Attorney Docket No. 826.1937

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Makoto NAKANISHI

Application No.: NEW

Group Art Unit: Not Yet Assigned

Filed: March 11, 2004

Examiner: Not Yet Assigned

For: MATRIX PROCESSING DEVICE IN SMP NODE DISTRIBUTED MEMORY THPE PARALLEL COMPUTER

SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN APPLICATION IN ACCORDANCE WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

Japanese Patent Application No(s). 2003-95720

Filed: March 31, 2003

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing date(s) as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: March 11, 2004

By: _____
J. Randall Beckers
Registration No. 30,358

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

JAPAN PATENT OFFICE

This is to certify that the annexed is a true copy of the following application as filed with this Office.

Date of Application: March 31, 2003

Application Number: Patent Application No. 2003-095720
[ST.10/C]: [JP2003-095720]

Applicant(s) : FUJITSU LIMITED

December 9, 2003

Commissioner,
JAPAN Patent Office Yasuo IMAI

Certificate No. P2003-3101648

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　３月３１日

出願番号
Application Number: 特願２００３－０９５７２０

[ST. 10/C]: [ＪＰ２００３－０９５７２０]

出願人
Applicant(s): 富士通株式会社

２００３年１２月　９日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３１０１６４８

【書類名】　特許願
【整理番号】　0350107
【提出日】　平成15年 3月31日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 17/12
【発明の名称】　ＳＭＰノード分散メモリ型並列計算機における行列処理装置
【請求項の数】　5
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　中西　誠
【特許出願人】
【識別番号】　000005223
【氏名又は名称】　富士通株式会社
【代理人】
【識別番号】　100074099
【住所又は居所】　東京都千代田区二番町８番地２０　二番町ビル３Ｆ
【弁理士】
【氏名又は名称】　大菅　義之
【電話番号】　03-3238-0031
【選任した代理人】
【識別番号】　100067987
【住所又は居所】　神奈川県横浜市鶴見区北寺尾７－２５－２８－５０３
【弁理士】
【氏名又は名称】　久木元　彰
【電話番号】　045-573-3683

【手数料の表示】

【予納台帳番号】　012542

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9705047

【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　　ＳＭＰノード分散メモリ型並列計算機における行列処理装置

【特許請求の範囲】

【請求項１】複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理方法であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの１巻き分を、該１巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第１の配置ステップと、

該１巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを１次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第２の配置ステップと、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にＬＵ分解するＬＵ分解ステップと、

ＬＵ分解されたブロックを用いて、行列の他のブロックを更新する更新ステップと、

を備えることを特徴とする行列処理方法を情報装置に実現させるプログラム。

【請求項２】前記ＬＵ分解は、再帰的手続きにより、各ノードの各プロセッサで並列的に行われることを特徴とする請求項１に記載のプログラム。

【請求項３】前記更新ステップにおいては、各ノードが、列ブロックを計算している間に、計算し終わった部分のデータであって、他のブロックの更新に必要なデータを該計算と平行して他のノードに転送することを特徴とする請求項１に記載のプログラム。

【請求項４】前記並列計算機は、ＳＭＰ（SymmetricMultiProcessor）を各ノードとするＳＭＰノード分散メモリ型並列計算機であることを特徴とする請求項１に記載のプログラム。

【請求項５】複数のプロセッサとメモリを含む複数のノードをネットワーク

で接続した並列計算機における行列処理装置であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの１巻き分を、該１巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第１の配置手段と、

該１巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離手段と、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを１次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第２の配置手段と、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にＬＵ分解するＬＵ分解手段と、

ＬＵ分解されたブロックを用いて、行列の他のブロックを更新する更新手段と、

を備えることを特徴とする行列処理装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ＳＭＰ（Symmetric　MultiProsessor）ノード分散メモリ型並列計算機における行列処理装置あるいは処理方法に関する。

【０００２】

【従来の技術】

ベクトルプロセッサをクロスバーで結合した並列計算機向けに開発した連立一次方程式の解法では、ブロックＬＵ分解の各ブロックを各ＰＥにサイクリックに配置してＬＵ分解を行っていた。ベクトルプロセッサではブロック幅を小さくしてもコストの高い行列積による更新部分の計算効率は非常に高かった。このためブロック幅１２程度でサイクリックな配置と見なして、まず、このブロックをＬＵ分解及び１つのＣＰＵで逐次的に計算してから、結果を部分的に分割して各プロセッサに転送して、行列積での更新を行っていた。

【０００３】

図２６は、スーパスカラ並列計算機用ＬＵ分解法のアルゴリズムを概略説明する図である。

配列Ａを外積形式のガウスの消去法をブロック化した方法でＬＵ分解する。ブロック幅ｄで分解する。

ｋ番目の処理で、更新部分$A^{(k)}$を次の計算で更新する。

$A^{(k)}=A^{(k)}-L2^{(k)}\times U2^{(k)}$・・・・・（１）

ｋ＋１番目の処理では、$A^{(k)}$を幅ｄで分解してｄだけ小さいマトリックスを同じ式で更新する。

$L2^{(k)}$、$U2^{(k)}$は以下の式で求める必要がある。

式（１）で更新を行う場合、

【０００４】

【数１】

$$\overline{A}^{(k)}=(L1^{(k)T},L2^{(k)T})^{T}\,U1^{(k)}$$

【０００５】

と分解し、$U2^{(k)}=L1^{(k)-1}U2^{(k)}$と更新する。

上記のブロック化されたＬＵ分解の方法は、特許文献１に記載されている。

そのほか、並列計算機で行列を計算する技術として特許文献２には、連立１次方程式の係数行列を外部記憶装置に格納する方式が、特許文献３には、ベクトル計算機における方式が、特許文献４には、多枢軸同時消去を行う方式が、特許文献５には、スパース行列の各要素の構成を並び替えて、縁付きブロック対角行列にしてからＬＵ分解を行う方法が記載されている。

【０００６】

【特許文献１】

特開２００２－１６３２４６号公報

【特許文献２】

特開平９－１７９８５１号公報

【特許文献３】

特開平１１－６６０４１号公報

【特許文献４】

特開平５－２０３４９号公報

【特許文献５】

特開平３－２２９３６３号公報

【０００７】

【発明が解決しようとする課題】

上記スーパスカラ並列計算機用ＬＵ分解の方法を単純に一つのノードをＳＭＰとする並列計算機システムで行うと以下の問題が発生する。

【０００８】

ＳＭＰノードでの行列積を効率的に行うためにはベクトル計算機で１２と設定していたブロック幅を１０００程度に増やす必要がある。

（１）この結果、ブロック毎にそれが各プロセッサにサイクリックに配置されていると見なして処理を行うと、行列積での更新の計算量がプロセッサ間で不均一である割合が大きくなり並列効率が著しく低下する。

（２）また、１ノードで計算する幅１０００程度のブロックのＬＵ分解は、ノード内でのみ計算すると、他のノードはアイドル状態となる。幅の大きさに比例して、このアイドル時間が増えるため、並列化効率が著しく低下する。

（３）ＳＭＰノードを構成するＣＰＵ数を増やすと計算能力の増加に対して、転送スピードが相対的に劣化しているため、従来の方法は転送量が約０．５n^2×１．５要素（ここでの要素は、行列の要素である）であったが、相対的に増えて見える。このため効率がかなり落ちる。

（１）～（３）までの劣化は全体で焼く２０～２５％の性能ダウンを引き起こす。

【０００９】

本発明の課題は、ＳＭＰノード分散メモリ型並列計算機で高速に行列を処理することの出来る装置あるいは方法を提供することである。

【００１０】

【課題を解決するための手段】

本発明の行列処理方法は、複数のプロセッサとメモリを含む複数のノードをネ

ットワークで接続した並列計算機における行列処理方法であって、ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの１巻き分を、該１巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第１の配置ステップと、該１巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを１次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第２の配置ステップと、該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にＬＵ分解するＬＵ分解ステップと、ＬＵ分解されたブロックを用いて、行列の他のブロックを更新する更新ステップとを備えることを特徴とする。

【００１１】

本発明によれば、各ノード間の計算負荷を分散し、並列化度を上げることが出来るので、より高速な行列処理が行える。また、演算と、データ転送を並列して行うことから、計算機の処理能力をデータ転送のスピードに制限されずに、向上することが出来る。

【００１２】

【発明の実施の形態】

本発明の実施形態においては、ブロック幅を大きくしても負荷バランスが完全に均一であり、１ＣＰＵで逐次計算していた部分をノード間で並列に処理する方式を提案する。

【００１３】

図１は、本発明の実施形態が適用されるＳＭＰノード分散メモリ型並列計算機の概略全体構成を示す図である。

図１（ａ）に示されるように、クロスバーネットワークにノード１～ノードＮが接続され、相互に通信できるようになっている。各ノードは、図１（ｂ）に示されるように相互結合網１０によって、メモリモジュール１１－１～１１－ｎ、及びプロセッサ１３－１～１３－ｍとキャッシュ１２－１～１２－ｍの組とが相互に結合され、通信可能となっている。データ通信用ハード（ＤＴＵ）１４は、

図１（ａ）のクロスバーネットワークに接続され、他のノードと通信可能となっている。

【００１４】

まず、比較的ブロック幅の小さなコラムブロックをノードにサイクリックに配置する。各ノードに一巻き分（サイクリックにコラムブロックを配置した場合の１回でサイクリックに配置される分）あるブロックを一つに束ねたものを一つに行列と見なす。これは行列を２次元目を均等に分割し、各ノードに分散配置した状態と見なすことが出来る。これを１次元目を均等に分割した配置に並列転送を利用して動的に変更する。ここで、１次元目を分割、２次元目を分割とは、行列を長方形あるいは正方形とした場合、横方向を縦の線で分割することを１次元目を分割すると言い、縦方向を横の線で分割することを２次元目を分割するという。このとき一番上の正方形部分は各ノードが重複して持つようにする。

【００１５】

この分散配置の変更でクロスバーネットワークを利用した並列転送が使え、転送量はノード数分の１となる。１次元目を均等に分割した配置に変更した配置で、ノード間通信を使って、このブロックのＬＵ分解を並列に行う。このとき並列化効率があがり、かつＳＭＰの性能を引き出せるようにするために、更にブロックに分解して再帰的なＬＵ分解を行う。

【００１６】

このブロックＬＵ分解が終了した時点で各ノードには対角ブロック部分の情報と１次元目を均等に分割した部分の情報があるため、これを利用して行ブロック部分を更新して、保持している列ブロック部分とで更新できる部分を更新する。更新時に隣のノードにこの情報を転送して、次の更新の準備を行う。この転送は計算と同時に行える。これらの操作を繰り返して全ての更新部分の更新を行う。

【００１７】

図２は、本発明の実施形態に従った全体の処理フローチャートである。

まず、ステップＳ１０において、最後の一巻きか否かを判断する。ステップＳ１０の判断がＹＥＳの場合には、ステップＳ１５に進む。ステップＳ１０の判断がＮＯの場合には、ステップＳ１１において、対象となる一巻き分のブロックを

結合したブロックを１次元目で分割した配置に並列転送を利用して変換する。このとき対角ブロックは全てのノードで共通に持つようにする。ステップＳ１２においては、１次元目を分割配置したブロックに関してＬＵ分解を行う。このときキャッシュの大きさを考慮したブロック幅までと、そのブロック幅より小さい部分の処理を再帰的な手続きで行う。ステップＳ１３では、ＬＵ分解した１次元目で分割配置されたブロックを並列転送をつかって元の２次元目を分割した配置に戻す。ステップＳ１４においては、この時点で各ノードには対角ブロックと残りをノード数に１次元目で分割した小ブロックが各ノードに割り付けられている。各ノードで共通に持っていた更新済みの対角ブロックを使ってブロック行を各ノードで更新する。このとき次の更新で必要となる列ブロックを隣のノードに計算と同時に転送する。ステップＳ１５では、最後の一巻きは各ノードに分割せずに冗長に配置して、同じ計算を行ってＬＵ分解を行う。各ノード部分に対応する部分をコピーバックする。そして、処理を終了する。

【００１８】

図３は、本発明の実施形態の一般概念図である。

図３に示されるように、行列を例えば、４等分して各ノードに分散配置する。各ノードは、列ブロックが割り当てられており、サイクリックな順序で処理する。このとき一巻き分を束ねて１つのブロックと見なす。これを対角ブロック部分を除き１次元目で分割し、通信を使って各ノードに再配置する。

【００１９】

図４及び図５は、比較的ブロック幅の小さなブロックをサイクリックに配置した状態を説明する図である。

図４及び図５に示すように、行列の一部の列ブロックを、更に小さい列ブロックに細分化し、各ノード（今の場合４つとしている）にサイクリックに割り当てる。このような配置の変更は、２次元目を分割されたブロックを１次元目を分割（対角ブロックは共通に保持）変更することになる。これはクロスバーネットワークの並列転送を利用して変更することが出来る。

【００２０】

これは、１巻きが結合されたブロックをメッシュに仮想的に分割したとき、対

角線方向のブロックの並び（１１、２２、３３、４４）、（１２、２３、３４、４１）、（１３、２４、３１、４２）、（１４、２１、３２、４３）の各組のブロックを各ノードに（二次元目の示すプロセッサから１次元目の示すプロセッサに転送する）並列転送することで実現できる。このとき、対角ブロック部分も一緒に送ることで対角ブロック部分は各ノードが共通に持つことができる充分な大きさで、転送はプロセッサ数分の１になる。

【００２１】

このように分散配置を変更した列ブロックに対するＬＵ分解を、各ノードに対角ブロックと残りの部分を均等に分割したものを配置して、ノード間通信及びノード間で同期を取りながら処理を行う。また、ノード内でのＬＵ分解の処理はスレッド並列化を行う。

【００２２】

スレッド並列化でのＬＵ分解がキャッシュ上で効率的に行えるように、２重構造の再帰的手続で行う。つまり、あるブロック幅までの大きさで一次の再帰手続で行い、それより小さい部分に関しては、スレッド並列化のために、各スレッドで、そのブロックを対角部分と残りの部分を並列処理するスレッド数で均等に分割した部分を合わせて連続な作業域にコピーして処理を行う。このことでキャッシュ上のデータを有効に利用する。

【００２３】

また、ノード間で共有している対角ブロック部分の計算はノード間で冗長に計算されてノード間のＬＵ分解の並列化効率が劣化する。ＬＵ分解を２重の再帰的手続きで行うことで、各ノード内でスレッドで並列計算するときのオーバヘッドを減らすことが出来る。

【００２４】

図６は、図４及び図５で配置されたブロックの更新処理を説明する図である。

図６の最も左のブロックは各ノードに対角ブロックを冗長に、かつ、残りのブロックを一次元目で均等に分割したものを作業域に配置したものである。あるノードでの状態と考える。最小ブロック幅まで１次の再帰手続きを行う。

【００２５】

最小ブロックのＬＵ分解が終わったら、この情報を使って、行ブロック及び更新部分の更新を更新する領域を均等に分割して、並列に更新する。

最小ブロック部分のＬＵ分解は、更に以下のように最小幅のブロックの対角部分を共通に、かつ、残り部分を均等に分割して、各スレッドの局所領域（キャッシュの大きさ程度）にコピーする。

【００２６】

この領域を使って、更に再帰的手続きでＬＵ分解を行う。ピボットを決めて、行の入れ替えを行うために各スレッドに、ピボットの相対的位置から、ノードでの相対位置、全体での位置に換算するための情報を保持しておく。

【００２７】

ピボットがスレッドの局所領域の対角部分内にあるときは、各スレッドで、独立に入れ替えを行える。

スレッドの対角ブロックを超えたときは、その位置が、以下の条件のときによって処理が異なる。

ａ）ピボットがノード間に分割配置したとき冗長に配置した対角ブロック内にあるとき。

【００２８】

このときは、ノード間で通信する必要はなく、各ノードで独立に処理できる。

ｂ）ピボットがノード間に分割配置した時冗長に配置した対角ブロックを超えたとき。

【００２９】

このときはスレッド間での最大値、つまりノードでの最大値を全ノードに通信して最大ピボットがどのノードに有るかを決定する。これが決まった後、最大ピボットを持つノードで行の入れ替えを行う。そのあと、入れ替えられた行（ピボット行）を他のノードに通信する。

【００３０】

このようなピボットの処理を行う。

２重構造を持つ再帰手続きでのＬＵ分解の二次のスレッド並列で行うＬＵ分解は、上記のピボット処理を行いながら、各スレッドの局所領域でＬＵ分解を並列

に行うことができる。

【００３１】

ピボットの入れ替えの履歴は共用メモリに各ノードに冗長に保持する。

図７は、再帰的なＬＵ分解の手順を説明する図である。

再帰的なＬＵ分解の手順は以下のようになる。

【００３２】

図７（ｂ）のレイアウトを考える。図７（ｂ）の対角ブロック部分がＬＵ分解できると、ＵはＬ１を使って、$U \leftarrow L1^{-1}U$、$C \leftarrow L \times U$と更新する。

再帰的手順は、ＬＵ分解する領域を前半と後半に分割し、分割した領域をＬＵ分解の対象と見なして、再帰的に行う方法である。ブロックの幅が、ある最小の幅より小さくなったとき、その幅に関しては従来通りのＬＵ分解を行う。

【００３３】

図６（ａ）は、領域を真ん中の太線で２分割し、その左側をＬＵ分解する過程で更に２分割したところである。太線で分割した左側は図６（ｂ）のレイアウトを当てはめられる。このレイアウトのＣの部分もＬＵ分解できたとき、太線から左側のＬＵ分解が終わる。

【００３４】

この左側の情報から、図６（ｂ）のレイアウトを全体にあてはめて、Ｃとなる右側の更新を行う。更新が終わったら、右側に図６（ｂ）のレイアウトを当てはめて同じようにＬＵ分解を行う。

・ブロックのＬＵ分解処理の後の行の入れ替えと行ブロックの更新及びrank　p updateでの更新

ノード間にブロックを再配置した状態でノード間通信及びスレッド並列を使ってＬＵ分解を並列に実行した後、各ノードには各ノードに共通に置かれた対角ブロックと残りの部分を均等に分割した部分のひとかけらがＬＵ分解された値を保持して残る。

【００３５】

各ノードでピボットの入れ替えの履歴の情報と対角ブロックの情報を使って、まず行の入れ替えを行う。その後、行ブロック部分の更新を行う。この後、対角

ブロックの残り部分を分割した列ブロック部分と更新された行ブロック部分を利用して更新部分を更新する。この計算と同時に更新に使う分割された列ブロック部分を全ノードで隣のノードに転送する。

【００３６】

この転送は、次の更新で必要な情報を計算と同時に送り、次の計算の前までに準備を行うためであり、転送を計算と同時に行うことで計算を効率よく続けることができる。

【００３７】

また、部分的な行列積の更新をスレッド数が多くても効率的に行えるように各スレッドで計算する行列積の更新領域が正方形に近くなる用に分割する。各ノードで更新を受け持つ更新領域は、正方形である。この領域の更新を各スレッドに分担して、かつ、性能劣化を引き起こさないようにすることを考える。

【００３８】

このため、更新領域をできるだけ正方形に近い形に分割する。このことで更新部分の２次元目の大きさがかなり大きく取れ、行列積の計算で繰り返し参照される部分の参照をキャッシュ上に保持して有効利用することが比較的できるようになる。

【００３９】

このために、以下の手順で行列積の更新の各スレッドでの分担を決めて並列計算する。

１）スレッドの総数＃ＴＨＲＤの平方根を求める。

２）この値が整数でないとき、これを切り上げてｎｒｏｗとする。

３）２次元目の分割数をｎｒｏｗとする。

４）１次元目の分割数をｎｃｏｌを以下の条件を満たす最小の整数を見つける。

ｎｃｏｌ×ｎｒｏｗ＞＝＃ＴＨＲＤ

５）if(ncol*nrow==#thrd)then

１次元目をｎｃｏｌ等分、２次元目をｎｒｏｗ等分ｎｃｏｌ＊ｎｒｏｗに分割して各スレッドに更新を並列実行させる。

else

１次元目をｎｃｏｌ等分、２次元目をｎｒｏｗ等分してｎｃｏｌ＊ｎｒｏｗに分割して（１、１）、（１、２）、（１、３）、・・・（２、１）、（２、２）、（２、３）・・・と＃ＴＨＲＤ個の部分を並列更新する。残りの領域は一般的に横に長い長方形となる。これを２次元目を均等に分割して全スレッドで負荷が均等になるように更新部分を分割して再度並列処理する。

endif

・ソルバー部分

図８は、対角部分以外の部分ブロックの更新について説明する図である。

【００４０】

ＬＵ分解された結果は、各ノードに分散配置された形で保存されている。各ノードには比較的ブロック幅の小さなブロックがＬＵ分解された状況で格納されている。

【００４１】

この幅の小さなブロックに関して前進代入、後退代入を行って次のブロックのある隣のノードに処理を渡す。このとき解を更新した部分を隣のノードに転送する。

【００４２】

実際の前進代入及び後退代入では細長いブロックで対角ブロック部分を除いた長方形部分を１次元目で均等にスレッド数で分割して並列更新を行う。

まず、一つスレッドでＬＤ×ＢＤ＝ＢＤを解く。

【００４３】

この情報を使って全スレッドで以下のようにＢを並列に更新する。

Ｂｉ＝Ｂｉ－Ｌｉ×ＢＤ

この１サイクルの更新で変更された部分を隣のノードに転送する。

【００４４】

前進代入が終わったら、今までの処理でノードに処理を渡してきたのとちょうど逆を辿るようにして後退代入を行う。

実際には、元の行列の各ノードに配置された部分をサイクリックに処理している。これは列ブロックを入れ替えて別の行列に変換していることに相当する。Ｌ

Ｕ分解の過程でピボットをとる列は未分解部分のどの列を対象にしてもよいことに由来する。

ＡＰＰ$^{-1}$ｘ＝ｂ→ｙ＝Ｐ$^{-1}$ｘと置いてｙについて解くことに相当する。解いたｙを並び変えることでｘを求めることが出来る。

【００４５】

図９～図１１は、行ブロックの更新処理を説明する図である。

列ブロックの計算が終わったら、今度計算された部分をもとの２次元目を分割した配置に戻す。ここで、２次元目を分割した形でのデータは各ノードに保持しておく。次に、行の入れ替え情報を元に、行の入れ替えを行ったあと、行ブロックを更新する。

【００４６】

各ノードに存在する列ブロックの部分を計算と同時に隣のノードにリング状に送ることで順次更新を進めていく。バッファをもう一つ持つことで可能となる。この領域には各ノードに対角ブロックを冗長に保持しているが、これも一緒に転送する。対角ブロック以外の部分のデータの量が多く、また、計算と同時に転送を行うので、転送時間は見えない。

【００４７】

図１０によれば、バッファＡからＢへのデータ転送を行う。次のタイミングではバッファＢからＡへのノードのリングに沿ってデータを送る。このようにしてスイッチしてデータ送る。更に、図１１において、更新が終わったら、列ブロックと行ブロックを除いた正方行列に対して大きさが縮小したもの対して同じ処理を繰り返す。

【００４８】

図１２～図２５は、本発明の実施形態のフローチャートである。

図１２及び図１３はサブルーチンｐLUのフローである。このサブルーチンは、呼び出しプログラムであり、各ノードで１つのプロセスを生成してから呼び出すことで並列に処理を行う。

【００４９】

まず、解くべき問題の大きさを、単位ブロック数をiblksunit、ノード数をnum

nordとして、n=iblksunit×numnord×ｍ（ｍは各ノードでの単位ブロック数）としたＬＵ分解を行う。各ノードに係数行列Ａの２次元目を均等に分割した共用メモリＡ（ｋ、ｎ／numnord）（ｋ＞＝ｎ）及び行の入れ替えの履歴を格納するip(n)を引数として受け取る。ステップＳ２０において、nonordにプロセス番号（１～ノード数）を設定し、numnordにノード数（全プロセス数）を設定する。ステップＳ２１において、各ノードでスレッドを生成し、nothrdにスレッド番号（１～スレッド数）及びnumthrdにスレッドの総数を設定する。ステップＳ２２において、ブロック幅の設定であるiblksmacro=iblksunit×numnord、繰り返し回数であるloop=n/(iblksunit×numthrd) -1を計算し、更に、i=1、lenbufmax=(n-iblksmacro)/numnord+iblksmacroを設定する。

【００５０】

ステップＳ２３において、wlul（lenbufmax, iblksmacro）、wlu2(lenbufmax, iblksmacro)、bufs(lenbufmax, iblksunit)、bufd(lenbufmax, iblksunit)の作業域を確保する。この領域をサブルーチンが実行の都度、実際の長さlenbufを計算して、必要な大きさだけ使う。

【００５１】

ステップＳ２４においては、i>=loopであるか否かを判断する。ステップＳ２４の判断がＹＥＳの場合には、ステップＳ３７に進む。ステップＳ２４の判断がＮＯの場合には、ステップＳ２５において、ノード間でバリア同期を取る。そして、ステップＳ２６において、lenblks=(n-i×iblksmacro)/numnord+iblksmacroを計算する。ステップＳ２７において、サブルーチンctobを呼び出し、各ノードにある幅iblksunitのｉ番目を対角ブロックと１次元目を均等分割した幅iblksmacroの部録を対角ブロックに結合し、ノードに持つ配置を変える。ステップＳ２８では、ノード間でバリア同期を取る。ステップＳ２９では、サブルーチンinterluを呼び出して、配列wlulに格納され、分散再配置された、ブロックをＬＵ分解する。行の入れ替えの情報は、is=(i-1)*iblksmacro+1, ie=i*iblksmacroとしてip(is:ie)に格納されている。

【００５２】

ステップＳ３０において、ノード間でバリア同期を取り、ステップＳ３１にお

いて、サブルーチンbtocを呼び出して、再配置されたブロックでＬＵ分解されたブロックを各ノードのもともと格納されていた場所に戻す。ステップＳ３２においてノード間でバリア同期を取り、ステップＳ３３において、サブルーチンexrwを呼び出して、行の入れ替え及び行ブロックの更新を行う。ステップＳ３４においては、ノード間でバリア同期を取り、ステップＳ３５において、サブルーチンmmcbtを呼び出して、各ノードにある列ブロックの部分（wlulに格納されている）と行ブロックの部部との行列積で更新する。計算と同時に列ブロック部分をプロセッサ間をリングに沿って転送し、次の更新の準備を行いながら更新する。ステップＳ３６においては、i=i+1として、ステップＳ２４に戻る。

【００５３】

ステップＳ３７では、ノード間でバリア同期を取り、ステップＳ３８において、生成したスレッドを消滅させる。ステップＳ３９において、サブルーチンfbluを呼んで、最後のブロックのＬＵ分解を行いながら更新する。ステップＳ４０において、ノード間でバリア同期を取り、処理を終了する。

【００５４】

図１４及び図１５は、サブルーチンctobのフローである。

ステップＳ４５において、Ａ（ｋ、ｎ／numnord）、wlul(lenblks,iblksmacro)、bufs（lenblks,iblksunit)、bufd（lenblks,iblksunit）を引数で受けて、各ノードのｉ番目の幅iblksunitのブロックをnumnord個束ねたものの対角ブロック行列部分より下の部分をnumnord個に分割したものと対角ブロックを加えたものとを各ノードに分散配置したものに転送を利用して配置換えする。

【００５５】

ステップＳ４６においては、nbase=(i-1)*iblksmacro（ｉは呼び出し元のメインループの繰り返し回数）、ibs=nbase+1、ibe=nbase+iblksmacro、len=(n-ibe)/numnord、nbase2d=(i-1)*iblksunit、ibs2d=nbase2d+1、ibe2d=ibs2d+iblksunitを計算する。ここで、送信データ数はlensend=(len+iblksmacro)*iblksunitである。ステップＳ４７においては、iy=1と設定し、ステップＳ４８において、iy>numnordか否かを判断する。ステップＳ４８の判断がＹＥＳの場合には、サブルーチンを抜ける。ステップＳ４８の判断がＮＯの場合には、ステップＳ４９にお

いて、送信する部分、受信する部分を決める。すなわち、idst=mod(nonord-1+iy-1,numnord)+1（送信先ノード番号）、isrs=mod(nonord-1+numnord-iy+1,numnord)+1（送信元ノード番号）を計算する。ステップＳ５０においては、各ノードで自分に割り付いている幅iblksunitの対角ブロック部分と、その下部分のブロックの１次元目をnumnordで分割した部分で、再配置した時保持する部分（転送先のノード数番目のもの）をバッファの下の部分に格納する。すなわち、bufd(1:iblksmacro,1:iblksunit)←A(ibs:ibe,ibs2d:ibe2d)、icps=ibe+(idst-1)+len+1、icpe=isps+len-1、bufd(iblksmacro+1:len+iblksmacro,1:iblksunit)←A（icps:icpe,ibs2d:ibe2d）を演算する。このコピーは１次元目をスレッド数に分割して各スレッドで並列に処理する。

【００５６】

ステップＳ５１では、全ノードで送受信を行う。すなわち、bufdの内容おidst番目のノードに送り、bufsに受信する。ステップＳ５２においては、送受信の完了を待つ。ステップＳ５３では、バリア同期を取り、ステップＳ５４において、wlulの対応位置に、isrs番目のノードから受けたデータを格納する。すなわち、icp2ds=(isrs-1)*iblksunit+1,icp2de=icp2ds+iblksunit-1、wlul(1:len+iblksmacr,,icp2ds:
icp2de)←bufs(1:len+iblksunit,1:blksunit)を演算する。すなわち、１次元目をスレッド数で分割して各スレッドで並列コピーする。ステップＳ５５でiy=iy+1とし、ステップＳ４８に戻る。

【００５７】

図１６及び図１７は、サブルーチンinterLUのフローである。

ステップＳ６０において、Ａ（ｋ、ｎ／numnord）、wlul(lenblks,iblksmacro)、wlumicro(ncash)を引数として受ける。ここで、wlumicroをＬ２キャッシュ（レベル２のキャッシュ）の大きさとし、各スレッドに確保されたものを受ける。wlulにＬＵブロック分解する幅iblksmacroのブロックで対角ブロックとその下位ブロックを１次元目でnumnord個に分割した一つが各ノードの領域に格納されている。ピボットの検索と行の入れ替えに関してノード間転送を使いながら並列にＬＵ分解する。本サブルーチンは、再帰的に呼び出される。呼び出しが深くなる

につれてＬＵ分解したときのブロック幅は小さくなる。このブロックをスレッド並列してＬＵ分解したとき、各スレッドで計算する部分がキャッシュの大きさ以下になるところで、ＬＵ分解をスレッド並列化する別のサブルーチンを呼び出す。

【００５８】

スレッド並列は対象となる比較的幅の小さなブロックをこのブロックの対角行列部分を各スレッドで重複して持ち、対角ブロックより下位の部分を１次元目をスレッド数で均等分割して各スレッド（ＣＰＵ）にキャッシュの大きさより小さな領域wlumicroで処理できるようにコピーして処理を行う。istmicroは小さなブロックの先頭位置であり、最初１に設定される。nwidthmicroは、小さなブロックの幅であり、最初は全体のブロック幅に設定される。iblksmicromaxは、小さなブロックの最大値であり、これ以上大きいときブロック幅を更に小さく（例えば、８０列に）する。nothrdはスレッド番号、numthrdはスレッド数、各ノードで重複して持つ１次元配列ip(n)に行の入れ替え情報を入れる。

【００５９】

ステップＳ６１では、nwidthmicro<=iblksmicromaxであるか否かを判断する。ステップＳ６１の判断がＹＥＳの場合には、ステップＳ６１において、iblksmicro=nwidthmicroとし、各ノードに分担した領域にある対角ブロックと分割したブロックが格納されているwlu(lenmacro, iblksmacro)のwlu(istmicro:lenmacro, istmicro:iblksmicro+iblksmicro-1)の部分に関して対角部分wlu(istmicro:istmicro+iblksmicro-1, istmicro:istmicro+iblksmicro-1)を対角ブロックとする。また、irest=istmicro+iblksmicroとし、wlu（irest:lenmacro, istmicro:istmicro+iblksmicro-1)を１次元目でスレッド数で均等分割したものを対角ブロックと結合して、各スレッド毎の領域wlumicroにコピーする。すなわち、lenmicro=(lenmaro-irest+numthrd)/numthrdとし、wlumicro(lenmicro+iblksmicro, iblksmicro)にコピーし、lenblksmicro=lenmicro+iblksmicroとする。そして、ステップＳ６３で、サブルーチンLUmicroを呼び出す。これにおいては、wlumicro(linmicro+iblksmicro, iblksmicro)を受け渡す。ステップＳ６４では、wlumicroに分割していた部分を、対角部分は１つのスレッドから、他の部分は各スレッドのwl

umicroからwluに元々あった部分に戻す。そして、サブルーチンを抜ける。

【００６０】

ステップＳ６１の判断がＮＯの場合には、ステップＳ６５において、nwidthmicro>=3*iblksmicromaxまたは、nwidthmicro<=2*iblksmicromaxか否かを判断する。ステップＳ６５の判断がＹＥＳの場合には、ステップＳ６６において、nwidthmicro2=nwidthmicro/2、istmicro2=istmicro+nwidthmicro2、nwidthmicro3=nwidthmicro-nwidthmicro2とし、ステップＳ６８に進む。ステップＳ６５の判断がＮＯの場合には、ステップＳ６７において、nwidthmicro2=nwidthmicro/3, istmicro2=istmicro+nwidthmicro2, nwidthmicro3=nwidthmicro-nwidthmicro2とし、ステップＳ６８に進む。ステップＳ６８では、istimicroは、そのまま、nwidthmicroとしてnwidthmicro2を渡してサブルーチンinterLUを呼び出す。

【００６１】

ステップＳ６９においては、wlu(istmicro:istmacro+nwidthmicro-1)の部分を更新する。これは、一つのスレッドで更新すれば充分である。これにwlu(istmicro:istmacro+nwidthmicro2-1, istmicro:istmacro+nwidthmicro2-1)の下三角行列の逆行列を左から乗じたもので更新する。ステップＳ７０においては、wlu(istmicro2:lenmacro, istmicro2:istmicro2+nwidthmicro3-1)をwlu(istmicro2:lenmacro, istmicro:istmicro2-1)×wlu(istmmicro:istmacro+nwidthmicro2-1, istmacro+
nwidthmicro2:istmacro+nwidthmicro-1)を引いて更新する。このとき、１次元目をスレッド数で均等に分割して並列計算する。ステップＳ７１においては、istmicroとして、istmicro2、nwidthmicroとしてnwidthmicro3を渡してサブルーチンinterLUを呼び出し、サブルーチンを終了する。

【００６２】

図１８及び図１９は、サブルーチンLUmicroのフローである。

ステップＳ７５において、Ａ（ｋ、ｎ／numnord）、wlul(lenblks, iblksmacro)、wlumicro(leniblksmicro, iblksmicro)を引数として受ける。ここで、wlumicroをＬ２キャッシュの大きさの各スレッドに確保されたものを受ける。本ルーチンでwlumicroに格納された部分のLU分解を行う。istは、LU分解するブロックの

先頭位置で最初は、１とされる。nwidthは、ブロック幅であり、最初は全体のブロック幅である。iblksmaxは、ブロック最大値（８程度）であり、これ以上小さくしない。wlumicroはスレッド毎に引数として渡される。

【００６３】

ステップＳ７６においては、nwidth<=iblksmaxか否かを判断する。ステップＳ７６の判断がＮＯの時は、ステップＳ８８に進む。ステップＳ７６の判断がＹＥＳの場合には、ステップＳ７７において、i=istとして、ステップＳ７８において、i<ist+nwidthか否かを判断する。ステップＳ７８の判断がＮＯの場合には、サブルーチンを抜ける。ステップＳ７８の判断がＹＥＳの場合には、ステップＳ７９において、各スレッドでｉ列目の絶対値最大の要素を見つけ、共用メモリ領域にスレッド番号順に格納する。ステップＳ８０においては、各ノードでのノード内の最大ピボットをこの中から見つけ、この要素とノード番号、位置をセットとして全ノードが各セットを持つように通信し、各ノードで全ノードでの最大ピボットを決定する。なお、各ノードで同じ方法で最大ピボットを決定する。

【００６４】

ステップＳ８１においては、このピボット位置が各ノードが持つ対角ブロックの中か判定する。ステップＳ８１の判断がＮＯの場合には、ステップＳ８５に進む。ステップＳ８１の判断がＹＥＳの場合には、ステップＳ８２において、最大ピボットの位置が各スレッドが重複して持つ対角ブロックの中かを判定する、ステップＳ８２の判断がＹＥＳの場合には、ステップＳ８３において、全ノードで保持する対角ブロック内での入れ替えで、かつ、全スレッドで重複して持つ対角部分内での入れ替えなので、スレッドで独立してピボットの入れ替えを行う。入れ替えた位置を配列ipに格納し、ステップＳ８６に進む。ステップＳ８２における判断がＮＯの場合には、ステップＳ８４において、各ノードで独立にピボットとを交換する。交換すべきピボット行を共用域に格納して、各スレッドの対角ブロック部分と入れ替える。入れ替えた位置を配列ipに格納し、ステップＳ８６に進む。

【００６５】

ステップＳ８５では、ノード間で通信して最大ピボットを有するノードから交

換すべき行ベクトルをコピーする。その後ピボット行を入れ替える。ステップＳ８６においては、行を更新し、ステップＳ８７において、ｉ列と行で更新部分を更新し、i=i+1として、ステップ７８に戻る。

【００６６】

ステップＳ８８においては、nwidth>=3*iblksmaxあるいは、nwidth<=2*iblksmaxであるか否かを判断する。ステップＳ８８の判断がＹＥＳの場合には、ステップＳ８９において、nwidth=nwidth/2、ist2=ist+nwidth2とし、ステップＳ９１に進む。ステップＳ８８の判断がＮＯの場合には、ステップＳ９０において、nwidth2=nwidth/3、ist2=ist+nwidth2、nwidth3=nwidth-nwidth2とし、ステップＳ９１に進む。ステップＳ９１では、istはそのまま、nwidthとしてnwidth2を引数として渡して、サブルーチンLUmicroを呼び出す。ステップＳ９２では、wlumicro(istmicro:istmacro+nwidth2-1,istmicro+nwidth2:istmicro+nwidthmicro-1)の部分を更新する。wlumicro(istmicro:istmacro+nwidth2-1,istmicro:istmacro+nwidth2-1)の下三角行列の逆行列を左から乗したもので更新する。ステップＳ９３では、wlumicro (istmicro2:lenmacro,istmicro2:istmicro2+nwidthmicro3-1)をwlumicro(istmicro2:lenmacro,istmicro:istmicro2-1)×wlumicro(istmicro:istmacro+nwidth2-1,ist+nwidth2:ist+nwidthmicro-1)を引いて更新する。ステップＳ９４においてはistとしてist2、nwidthとしてnwidth3を引数として受け渡して、サブルーチンLUmicroを呼び出して、サブルーチンを抜ける。

【００６７】

図２０は、サブルーチンbtocのフローである。

ステップＳ１００において、Ａ（ｋ、ｎ／numnord）、wlul (lenblks,iblksmacro)、bufs(lenblks,iblksunit)、bufd(lenblks,iblksunit)を引数で受けて、各ノードのｉ番目の幅iblksunitのブロックをnumnord個束ねたものの対角ブロック行列部分iblksmacro×iblksmacroより下の部分をnumnord個に分割したものと対角ブロックを加えたものを各ノードに分散配置したものに転送を利用して配置を変える。

【００６８】

ステップＳ１０１では、nbase=(i-1)*iblksmacro（ｉは呼び出しもとのメイン

ループの繰り返し回数）、ibs=nbase+1、ibe=nbase+iblksmacro、len=(n-ibe)/numnord、nbase2d=(i-1)*iblksunit、ibs2d=nbase2d+1、ibe2d=ibs2d+iblksunitとし、送信データ数は、lensend=(len+iblksmacro)*iblksunitとする。

【００６９】

ステップＳ１０２において、iy=1とし、ステップＳ１０３において、iy>numnordか否かを判断する。ステップＳ１０３の判断がＹＥＳの場合、サブルーチンを抜ける。ステップＳ１０３の判断がＮＯの場合には、ステップＳ１０４において、送信する部分、受信する部分を決める。すなわち、idst=mod(nonord-1+iy-1,numnord)+1、isrs=mod(nonord-1+numnord-iy+1,numnord)+1とする。ステップＳ１０５においては、計算結果が格納されているwlulから元の位置に配置を戻すための送信のためにバッファに格納する。idst番目のノードに対応部分を送る。すなわち、icp2ds=(idst-1)*iblksunit+1、icp2de=icp2ds+iblksunit-1、bufd(1:len+iblksunit,1:iblksunit)←wlul(1:len+iblksmacro,icp2ds:icp2de)とする。１次元目をスレッド数で分割して各スレッドで並列コピーする。

【００７０】

ステップＳ１０６では、全ノードで送受信する。bufdの内容をidst番目のノードに送り、bufsに受信する。ステップＳ１０７で送受信の完了を待ち、ステップＳ１０８において、バリア同期を取る。ステップＳ１０９では、各ノードで自分に割り付いている幅iblksunitの対角ブロック部分と、その下の部分のブロックの１次元目をnumnordで分割した部分で再配置したときの部分（転送先のノード数番目のもの）を元々あった部分に格納する。A（ibs:ibe,ibs2d:ibd2d)←bufs(1:iblksmacro,1:iblksunit)、icps=ibe+(isrs-1)*len+1、icpe=isps+len-1、A(icps:icpe,ibs2d:ibe2d)←bufs(iblksmacro+1:len+iblksmacro,1:iblksunit)とする。このコピーは１次元目をスレッド数に分割して各スレッドで列毎に処理する。

【００７１】

ステップＳ１１０においては、iy=iy+1として、ステップＳ１０３に戻る。

図２１は、サブルーチンexrwのフローである。

このサブルーチンは、行の入れ替え及び行ブロックの更新を行うものである。

【００７２】

ステップＳ１１５においては、Ａ（ｋ、ｎ／numnord）、wlul（lenblks、iblksmacro）を引数として受ける。wlul（1:iblksmacro,1:iblksmacro）には、ＬＵ分解された対角部分を全ノードが重複して持っている。nbdiag=(i-1)*iblksmacroとする。ｉは呼び出し元のサブルーチンｐＬＵのメインループの繰り返し回数である。また、ピボットの入れ替えの情報が、ip(nbdiag+1:nbdiag+iblksmacro)に格納されている。

【００７３】

ステップＳ１１６では、nbase=i*iblksunit（ｉは呼び出しもとのサブルーチンｐＬＵのメインループの繰り返し回数）、irows=nbase+1、irowe=n/numnord、len=(irowe-irows+1)/numthrd、is=nbase+(nothrd-1)*len+1、ie=min(irowe,is+len-1)とする。ステップＳ１１７では、ix=isとする。

【００７４】

ステップＳ１１８では、is<=ieであるか否かを判断する。ステップＳ１１８の判断がＮＯの場合には、ステップＳ１２５に進む。ステップＳ１１８の判断がＹＥＳの場合には、ステップＳ１１９において、nbdiag=(i-1)*iblksmacro、j=nbdag+1として、ステップＳ１２０において、j<=nbdiag+iblksmacroであるか否かを判断する。ステップＳ１２０の判断がＮＯの場合には、ステップＳ１２４に進む。ステップＳ１２０の判断がＹＥＳの場合には、ステップＳ１２１において、ip(j)>jか否かを判断する。ステップＳ１２１の判断がＮＯの場合には、ステップＳ１２３に進む。ステップＳ１２１の判断がＹＥＳの場合には、ステップＳ１２２において、Ａ（j、ix）とＡ（ip(j),ix)を入れ替えて、ステップＳ１２３に進む。ステップＳ１２３においては、j=j+1として、ステップＳ１２０に戻る。

【００７５】

ステップＳ１２４においては、ix=ix+1とし、ステップＳ１１８に戻る。

ステップＳ１２５においては、バリア同期（全ノード、全スレッド）を取る。ステップＳ１２６においては、Ａ（nbdiag+1:nbdiag+iblksmacro,is:ie)←TRL（wlul（i:iblksmacro,1:iblksmacro))$^{-1}$×Ａ（nbdiag+1:nbdiag+iblksmacro,is:ie）を全ノード、全スレッドで更新する。ここで、TRL（Ｂ）は、行列Ｂの下三角

部分を示す。ステップＳ１２７では、バリア同期（全ノード、全スレッド）を取って、サブルーチンを抜ける。

【００７６】

図２２及び図２３は、サブルーチンmmcbtのフローである。

ステップＳ１３０において、Ａ（ｋ、ｎ／numnord）、wlul（lenblks、iblksmacro）、wlu2（lenblks,iblksmacro）を引数として受ける。wlulに、ブロック幅iblksmacroのブロックをＬＵ分解した結果で、対角ブロックとその下位ブロックを１次元目でnumnord個に分割した一つが格納されている。分割した順にノード番号に対応し、ノードに再配置される。これをノードのリングに沿って転送しながら（計算と同時に行う）行列積を行いながら更新する。計算の裏で性能に影響を与えないので計算に直接使用しない対角ブロック部分も一緒に送る。

【００７７】

ステップＳ１３１では、nbase=(i-1)*iblksmacro（ｉは呼び出しもとのサブルーチンｐＬＵのメインループの繰り返し回数）、ibs=nbase+1、ibe=nbase+iblksmacro、len=(n-ibe)/numnord、nbase2d=(i-1)*iblksunit、ibs2d=nbase2d+1、ibe2d=ibs2d+iblksunit、n2d=n/numnord、lensend=len+iblksmacroとし、送信データ数は、nwlen=lensend*iblksmacroとする。

【００７８】

ステップＳ１３２において、iy=1（初期値を設定）、idst=mod(nonord,numnord)+1（送り先ノード番号（隣ノード））、isrs=mod(nonord-1+numnord-1,numnord)+1（発信元ノード番号）、ibp=idstとする。

【００７９】

ステップＳ１３３において、iy>numnordであるか否かを判断する。ステップＳ１３３の判断がＹＥＳの場合には、サブルーチンを抜ける。ステップＳ１３３の判断がＮＯの場合には、ステップＳ１３４において、iy=1か否かを判断する。ステップＳ１３４の判断がＹＥＳの場合には、ステップＳ１３６に進む。ステップＳ１３４の判断がＮＯの場合には、ステップＳ１３５において送受信の官僚を待つ。ステップＳ１３６では、iy=numnord（奇数の最後）であるか否かを判断する。ステップＳ１３６の判断がＹＥＳの場合には、ステップＳ１３８に進む。ステ

ップＳ１３６の判断がＮＯの場合には、ステップＳ１３７において、送受信を行う。wlulの内容を（対角ブロックも含めて）隣のノード（ノード番号idst）に送る。かつ、wlu2に（ノード番号isrsから）送られてくるデータを格納する。送受信データ長はnwlenとする。

【００８０】

ステップＳ１３８において、wlulのデータを使った更新のポジションを計算する。ibp=mod(ibp-1+numnord-1,numnord)+1、ncptr=nbe+(ibp-1)*len+1（１次元目の開始位置）とする。ステップＳ１３９では、行列積を計算するサブルーチンpmmを呼び出す。このときwlulを引き渡す。ステップＳ１４０において、iy=numnord（最後の処理が終わった）か否かを判断する。ステップＳ１４０の判断がＹＥＳの場合には、サブルーチンを抜ける。ステップＳ１４０の判断がＮＯの場合には、ステップＳ１４１において、行列積演算と同時に行っている送受信の完了を待つ。ステップＳ１４２において、iy=numnord-1（偶数の最後）であるか否かを判断する。ステップＳ１４２の判断がＮＯの場合には、ステップＳ１４４に進む。ステップＳ１４２の判断がＮＯの場合には、ステップＳ１４３において、送受信を行う。すなわち、wlu2の内容を（対角ブロックも含めて）隣のノード（ノード番号idst）に送る。かつ、wlulに（ノード番号isrsから）送られてくるデータを格納する。送受信データ長はnwlenとする。

【００８１】

ステップＳ１４４では、wlu2のデータを使った更新のポジションを計算する。すなわち、ibp=mod(ibp-1+numnord-1,numnord)+1、ncptr=nbe+(ibp-1)*len+1（１次元目の開始位置）とする。

【００８２】

ステップＳ１４５では、行列積を計算するサブルーチンｐmmを呼び出す。このとき、wlu2を引き渡す。ステップＳ１４６において、iy=iy+2と、２を加えて、ステップＳ１３３に戻る。

【００８３】

図２４は、サブルーチンpmmのフローである。

ステップＳ１５０において、Ａ（ｋ、ｎ／numnord）、wlul（lenblks、iblksm

acro）、もしくは、wlu2（lenblks,iblksmacro）をwlux(lenblks,iblksmacro)に受ける。呼び出し元から渡された１次元目の開始位置ncptrを使って正方形の領域を更新する。is2d=i*iblksunit+1、ie2d=n/numnord、len=ie2d-is2d+1、isld=ncptr、ield=nptr+len-1（ｉはサブルーチンｐＬＵの繰り返し数）、Ａ（isld:ield,is2d:ie2d）＝Ａ（isld:ield,is2d:ie2d)-wlu(iblksmacro+1:iblksmacro+len,1:iblksmacro)×Ａ（isld-iblksmacro:isld-1,is2d:ie2d)（式１）とする。

【００８４】

ステップＳ１５１において、並列に処理するスレッド数の平方根を求めて切り上げる。numroot=int(sqrt(numthrd))、もし、sqrt(numthrd)-numrootが０でないなら、numroot=numroot+1とする。ここで、ｉｎｔは小数点以下切り捨て、ｓｑｒｔは、平方根である。ステップＳ１５２において、m1=numroot、m2=numroot、mx=m1とする。ステップＳ１５３において、m1=mx、mx=mx-1、mm=mx×m2とする。ステップＳ１５４において、mm<numthrdであるか否かを判断する。ステップＳ１５４の判断がＮＯの場合には、ステップＳ１５３に戻る。ステップＳ１５４の判断がＹＥＳの場合には、ステップＳ１５５において、更新する領域を１次元目をｍ１等分する。２次元目をｍ２等分して、ｍ１×ｍ２個の矩形にする。そのうち、numthrd個を各スレッドに割り当てて、（式１）の対応部分を並列に計算する。（１，１）、（１，２）、・・・（１，ｍ２）、（２，１）・・・・と２次元目の方向に順番にスレッドを対応付けていく。

【００８５】

ステップＳ１５６において、m1*m2-numthrd>0か否かを判断する。ステップＳ１５６の判断がＹＥＳの場合には、ステップＳ１５８に進む。ステップＳ１５６の判断がＮＯの場合には、ステップＳ１５７において、残りの矩形は最後の矩形の最後の行、１行の最後からm1*m2-numthrd個が更新されずに残っている。この矩形を結合して１つの矩形と考え、２次元目をスレッド数numthrdで分割して（式１）の対応部分を並列に計算する。そして、ステップＳ１５８において、バリア同期（スレッド間）をとって、サブルーチンを抜ける。

【００８６】

図２５は、サブルーチンfbluのフローである。

ステップＳ１６０において、Ａ（ｋ、ｎ／numnord）、wlul（iblksmacro、iblksmacro）、bufs（iblksmacro、iblksunit）、bufd（iblksmacro、iblksunit）を引数で受けて、各ノードの幅iblksunitの最後のブロックをnumnord個束ねたものを各ノードで重複して持つように利用不足部分を各ノードに送る。各ノードがiblksmacro×iblksmacroのブロックを重複して持った後、各ノードで同じ行列に対してＬＵ分解を行う。ＬＵ分解が完了したら、各ノードに配置されていた部分をコピーバックする。

【００８７】

ステップＳ１６１では、nbase=n-iblksmacro、ibs=nbase+1、ibe=n、len=iblksmacro、nbase2d=(i-1)*iblksunit、ibs2d=n/numnord-iblksunit+1、ibe2d=n/numnordとし、送信データ数はlensend=iblksmacro*iblksunitとし、iy=1とする。

【００８８】

ステップＳ１６２においては、バッファへのコピーを行う。すなわち、bufd(1:iblksmacro,1:iblksunit)←A(ibs:ibe,ibs2d:ibe2d)とする。ステップＳ１６３においては、iy>numnordか否かを判断する。ステップＳ１６３の判断がＹＥＳの場合には、ステップＳ１７０に進む。ステップＳ１６３の判断がＮＯの場合には、ステップＳ１６４において、送信する部分、受信する部分を決定する。すなわち、idst=mod(nonord-1+iy-1,numnord)+1、isrs=mod(nonord-1+numnord-iy+1,numnord)+1とする。ステップＳ１６５では、全ノードで送受信する。bufdの内容をidst番目のノードに送る。ステップＳ１６６においては、bufsにデータを受信し、送受信の完了を待つ。ステップＳ１６７において、バリア同期を取り、ステップＳ１６８において、wlulの対応位置にisrs番目のノードから来たデータを格納する。icp2ds=(isrs-1)*iblksunit+1、icp2de=icp2ds+iblksunit-1、wlu(1:iblksmacro,icp2ds:icp2de)←bufs(1:iblksunit,1:iblksunit)とする。ステップＳ１６９において、iy=iy+1とし、ステップＳ１６３に戻る。

【００８９】

ステップＳ１７０では、バリア同期をとり、ステップＳ１７１では、wlulの上でiblksmacro×iblksmacroのＬＵ分解を各ノードで重複して行う。行交換の情報は、ipに格納する。ＬＵ分解が終了したら、自ノード分を最後のブロックにコピ

ーバックする。すなわち、is=(nonord-1)*iblksunit+1、ie=is+iblksunit-1、A(ibs:ibe, ibs2d:ibe2d)←wlul(1:iblksmacro, is:ie)として、サブルーチンを抜ける。

（付記１）複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理方法であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの１巻き分を、該１巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第１の配置ステップと、

該１巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを１次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第２の配置ステップと、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にＬＵ分解するＬＵ分解ステップと、

ＬＵ分解されたブロックを用いて、行列の他のブロックを更新する更新ステップと、

を備えることを特徴とする行列処理方法を情報装置に実現させるプログラム。

【００９０】

（付記２）前記ＬＵ分解は、再帰的手続きにより、各ノードの各プロセッサで並列的に行われることを特徴とする付記１に記載のプログラム。

（付記３）前記更新ステップにおいては、各ノードが、列ブロックを計算している間に、計算し終わった部分のデータであって、他のブロックの更新に必要なデータを該計算と平行して他のノードに転送することを特徴とする付記１に記載のプログラム。

【００９１】

（付記４）前記並列計算機は、ＳＭＰ（SymmetricMultiProcessor）を各ノードとするＳＭＰノード分散メモリ型並列計算機であることを特徴とする付記１

に記載のプログラム。

【００９２】

（付記５）複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理装置であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの１巻き分を、該１巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第１の配置手段と、

該１巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離手段と、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを１次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第２の配置手段と、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にＬＵ分解するＬＵ分解手段と、

ＬＵ分解されたブロックを用いて、行列の他のブロックを更新する更新手段と、

を備えることを特徴とする行列処理装置。

【００９３】

（付記６）複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理方法であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの１巻き分を、該１巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第１の配置ステップと、

該１巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを１次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第２の配置ステップと、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノー

ドで並列にＬＵ分解するＬＵ分解ステップと、

ＬＵ分解されたブロックを用いて、行列の他のブロックを更新する更新ステップと、

を備えることを特徴とする行列処理方法。

【００９４】

（付記７）複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理方法であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの１巻き分を、該１巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第１の配置ステップと、

該１巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを１次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第２の配置ステップと、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にＬＵ分解するＬＵ分解ステップと、

ＬＵ分解されたブロックを用いて、行列の他のブロックを更新する更新ステップと、

を備えることを特徴とする行列処理方法を情報装置に実現させるプログラムを格納する、情報装置読み取り可能な記録媒体。

【００９５】

【発明の効果】

ブロックを動的に１次元目の分割にして処理し、分解した後の各ノードの情報を使って更新し、転送は計算と同時に行える。このため更新部分は負荷はノード間で完全に均等になり、転送量はノード数分の１に削減できる。

【００９６】

ブロック幅を大きくすると負荷のバランスが崩れる従来の方法に対し負荷が均

等になるため並列化効率が１０％程度向上する。また、転送量が減ることで３％程度の並列化率の向上に寄与でき、転送スピードがＳＭＰノードの計算性能に比べて遅くなっても影響は受けにくい。

【００９７】

ブロック部分のＬＵ分解をノード間で並列計算することによって、ブロック幅を大きくしたとき並列化出来ない部分の割合が増加するため並列化効率が落ちる部分をキャンセルできて約１０％の並列化効率の向上が見込める。また、ブロックＬＵ分解を、ミクロなブロックをベースにした再帰的プログラミングを使うことで、対角ブロックも含めてＳＭＰの並列化ができてＳＭＰでの並列処理での性能劣化を抑えることができる。

【図面の簡単な説明】

【図１】

本発明の実施形態が適用されるＳＭＰノード分散メモリ型並列計算機の概略全体構成を示す図である。

【図２】

本発明の実施形態に従った全体の処理フローチャートである。

【図３】

本発明の実施形態の一般概念図である。

【図４】

比較的ブロック幅の小さなブロックをサイクリックに配置した状態を説明する図（その１）である。

【図５】

比較的ブロック幅の小さなブロックをサイクリックに配置した状態を説明する図（その２）である。

【図６】

図４及び図５で配置されたブロックの更新処理を説明する図である。

【図７】

再帰的なＬＵ分解の手順を説明する図である。

【図８】

対角部分以外の部分ブロックの更新について説明する図である。

【図９】

行ブロックの更新処理を説明する図（その１）である。

【図１０】

行ブロックの更新処理を説明する図（その２）である。

【図１１】

行ブロックの更新処理を説明する図（その３）である。

【図１２】

本発明の実施形態のフローチャート（その１）である。

【図１３】

本発明の実施形態のフローチャート（その２）である。

【図１４】

本発明の実施形態のフローチャート（その３）である。

【図１５】

本発明の実施形態のフローチャート（その４）である。

【図１６】

本発明の実施形態のフローチャート（その５）である。

【図１７】

本発明の実施形態のフローチャート（その６）である。

【図１８】

本発明の実施形態のフローチャート（その７）である。

【図１９】

本発明の実施形態のフローチャート（その８）である。

【図２０】

本発明の実施形態のフローチャート（その９）である。

【図２１】

本発明の実施形態のフローチャート（その１０）である。

【図２２】

本発明の実施形態のフローチャート（その１１）である。

【図２３】

本発明の実施形態のフローチャート（その１２）である。

【図２４】

本発明の実施形態のフローチャート（その１３）である。

【図２５】

本発明の実施形態のフローチャート（その１４）である。

【図２６】

スーパスカラ並列計算機用ＬＵ分解法のアルゴリズムを概略説明する図である。

【符号の説明】

１０　　相互結合網（バス）

１１－１～１１－ｎ　　メモリモジュール

１２－１～１２－ｍ　　キャッシュ

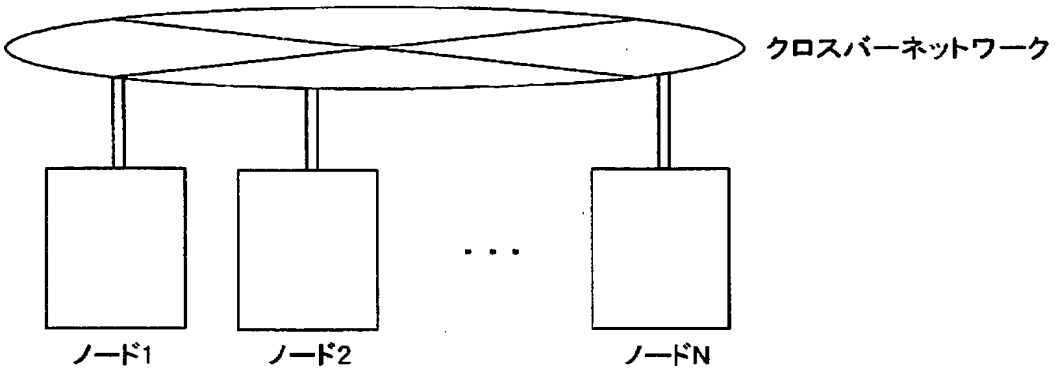
１３－１～１３－ｍ　　プロセッサ

１４　　データ通信用ハード（ＤＴＵ）

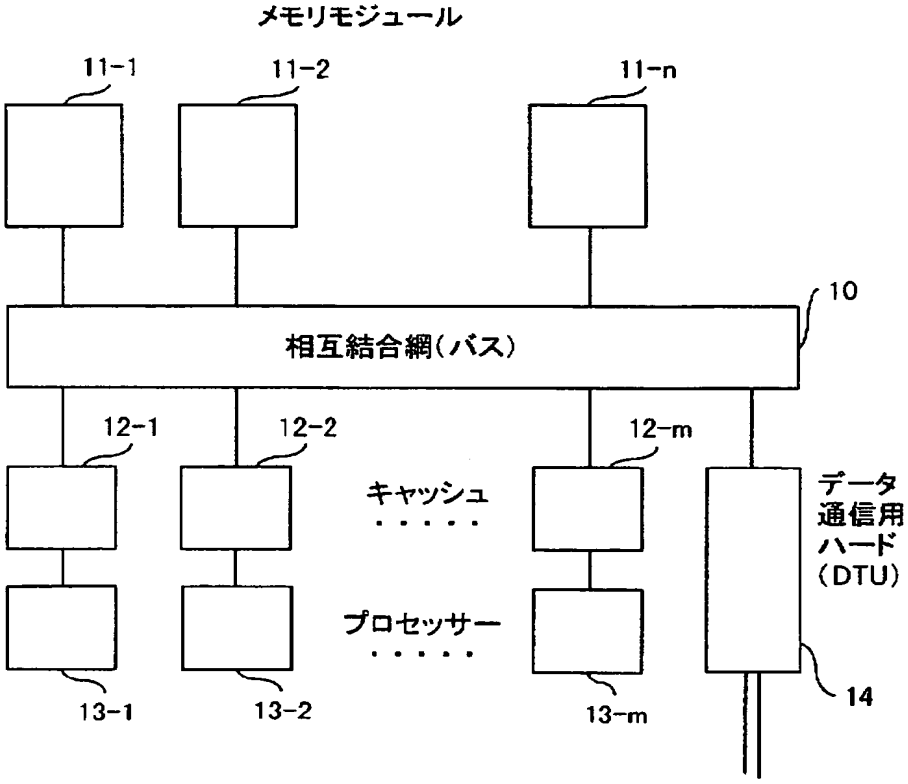
【書類名】　図面

【図１】

本発明の実施形態が適用されるSMPノード分散メモリ型並列計算機の概略全体構成を示す図



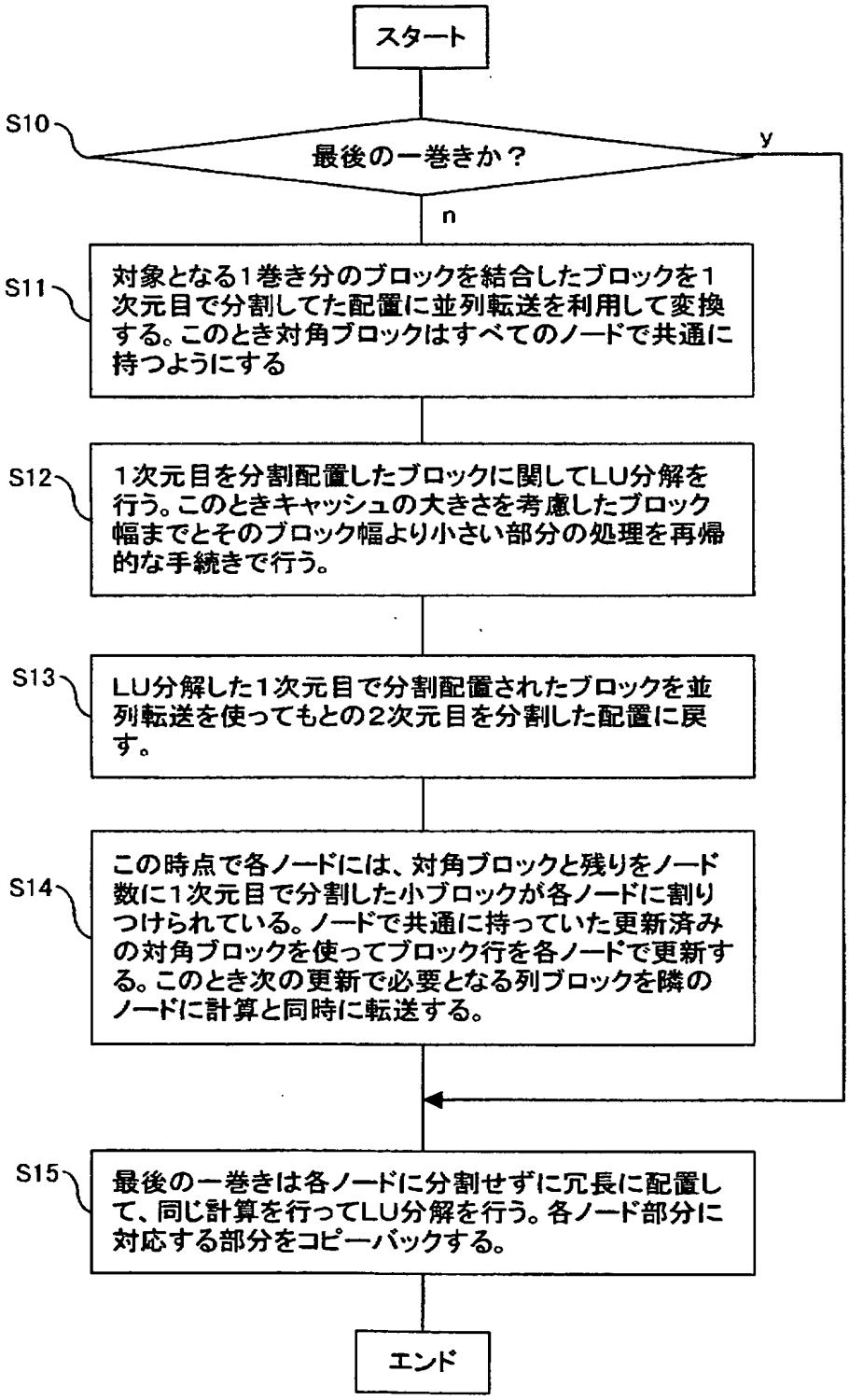
（ａ）



（ｂ）

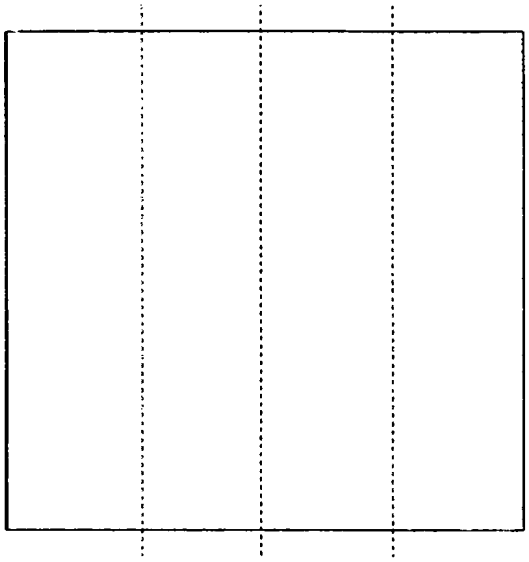
【図２】

本発明の実施形態に従った全体の処理フローチャート



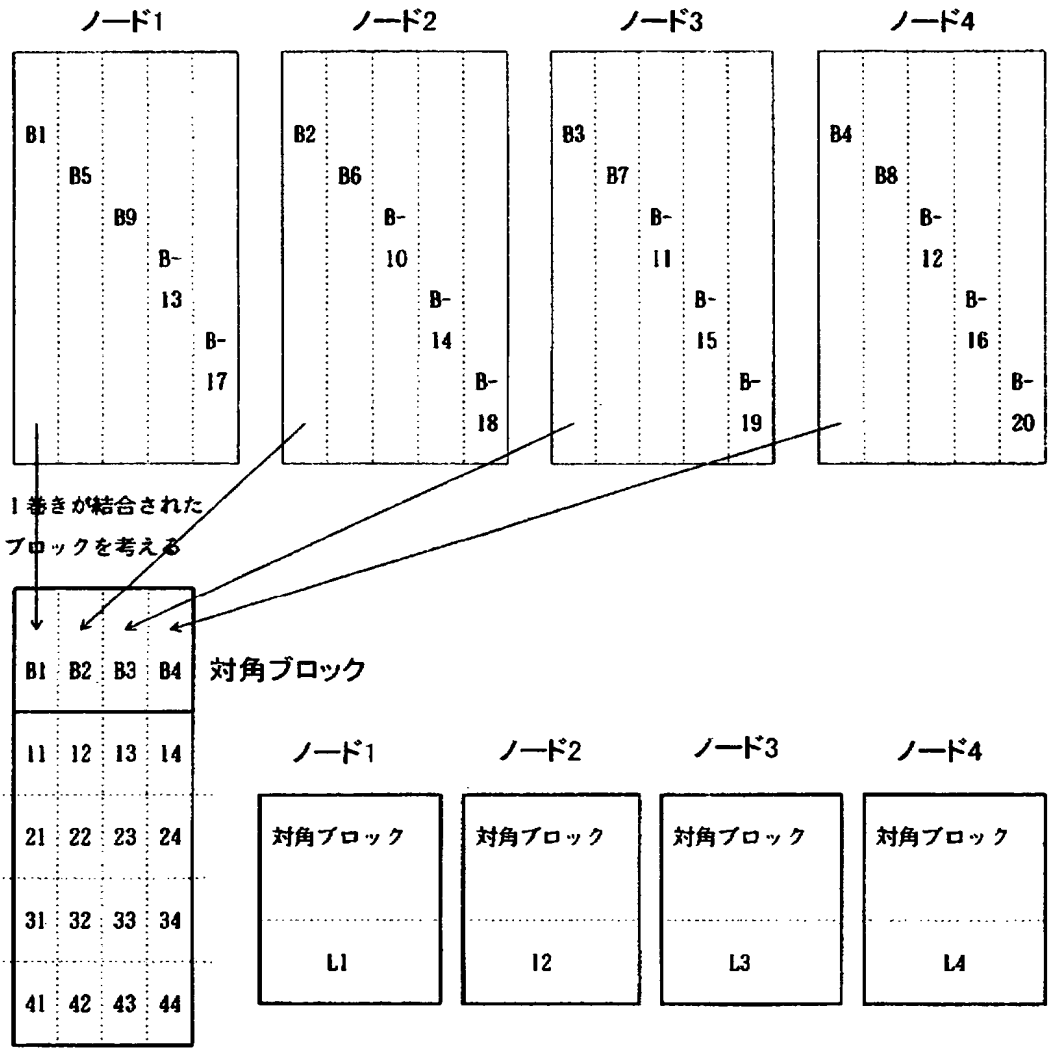
【図３】

本発明の実施形態の一般概念図



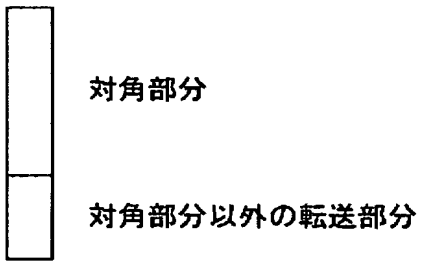
【図４】

比較的ブロック幅の小さなブロックをサイクリックに配置した状態を説明する図（その１）



【図５】

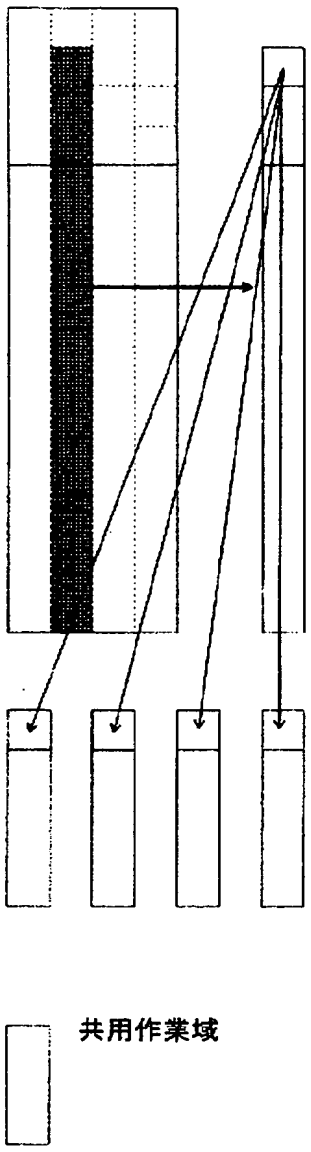
比較的ブロック幅の小さなブロックをサイクリックに配置した状態を説明する図（その２）



転送で使う作業領域

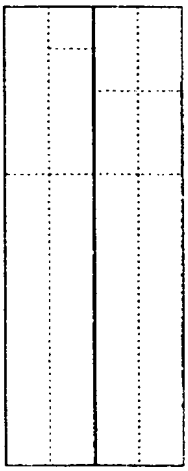
【図６】

図４及び図５で配置されたブロックの更新処理を説明する図

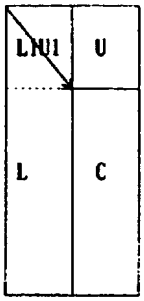


【図７】

再帰的なLU分解の手順を説明する図



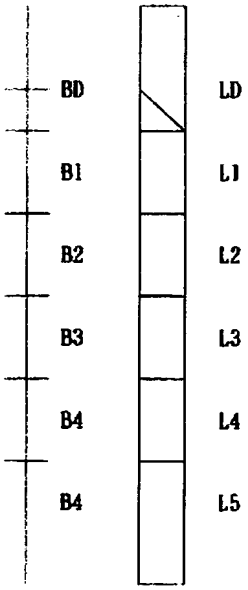
（ａ）



（ｂ）

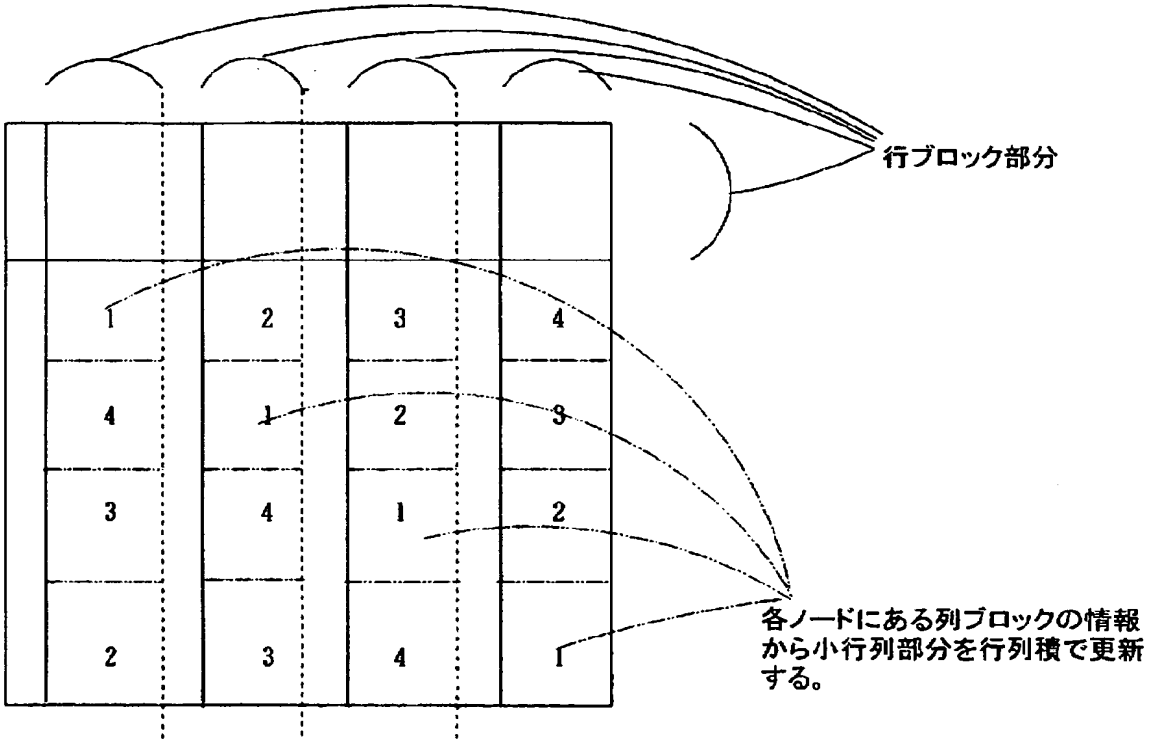
【図８】

対角部分以外の部分ブロックの更新について説明する図



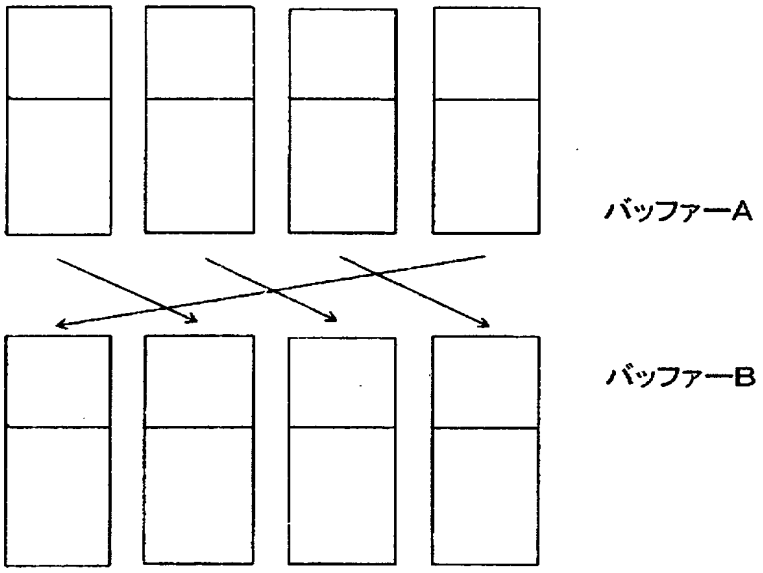
【図９】

行ブロックの更新処理を説明する図（その１）



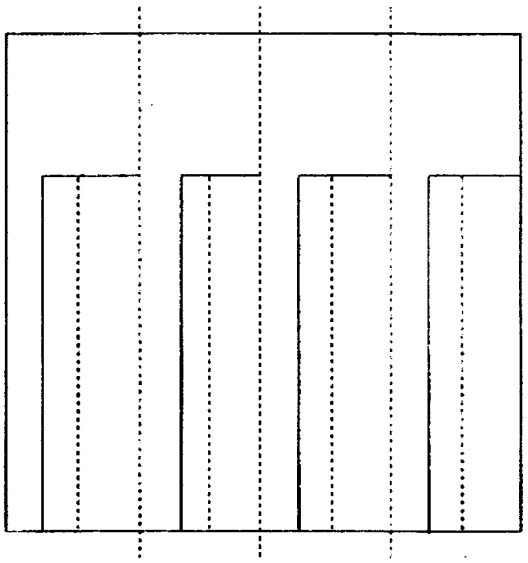
【図１０】

行ブロックの更新処理を説明する図(その2)



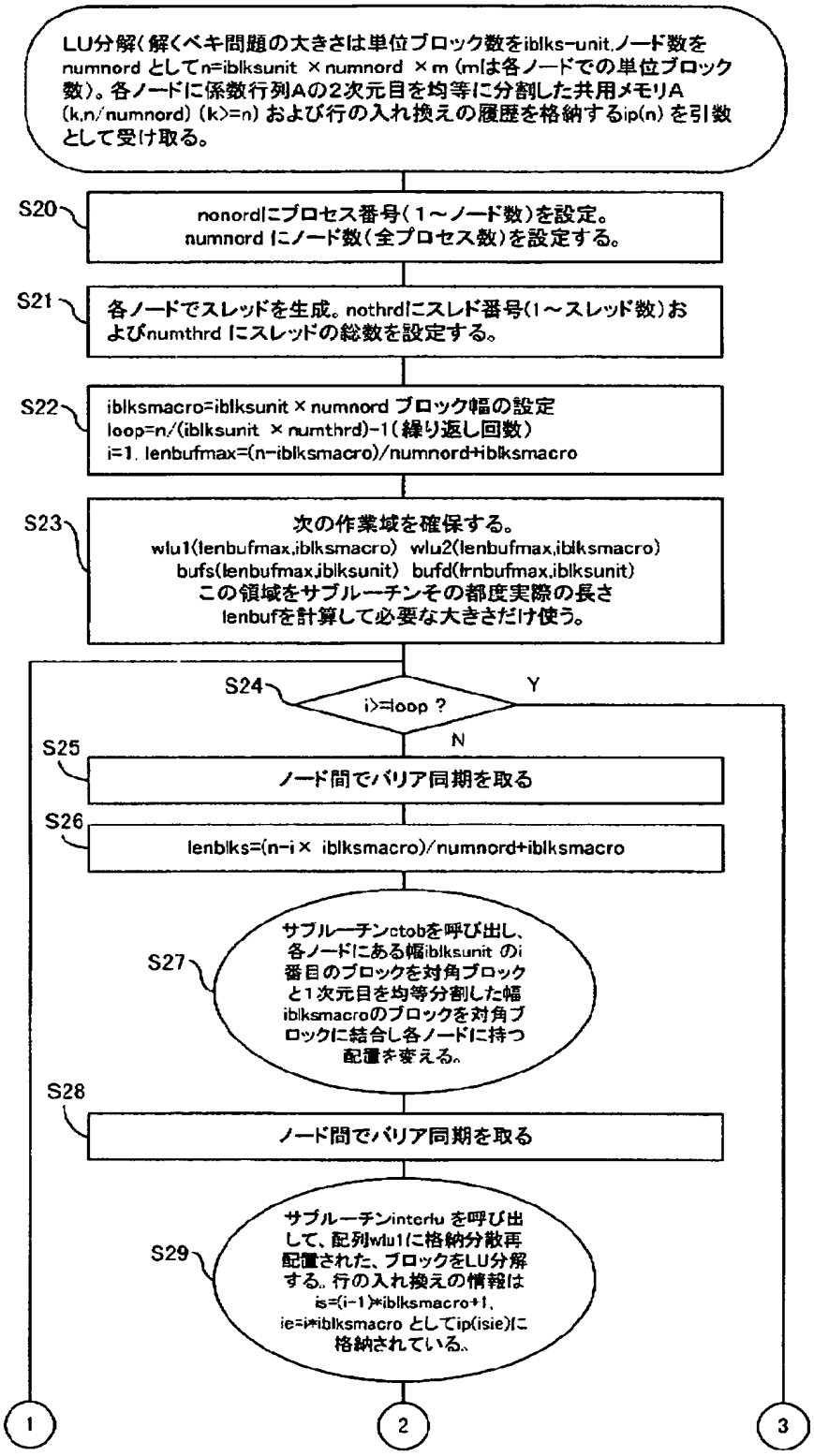
【図１１】

行ブロックの更新処理を説明する図（その3）



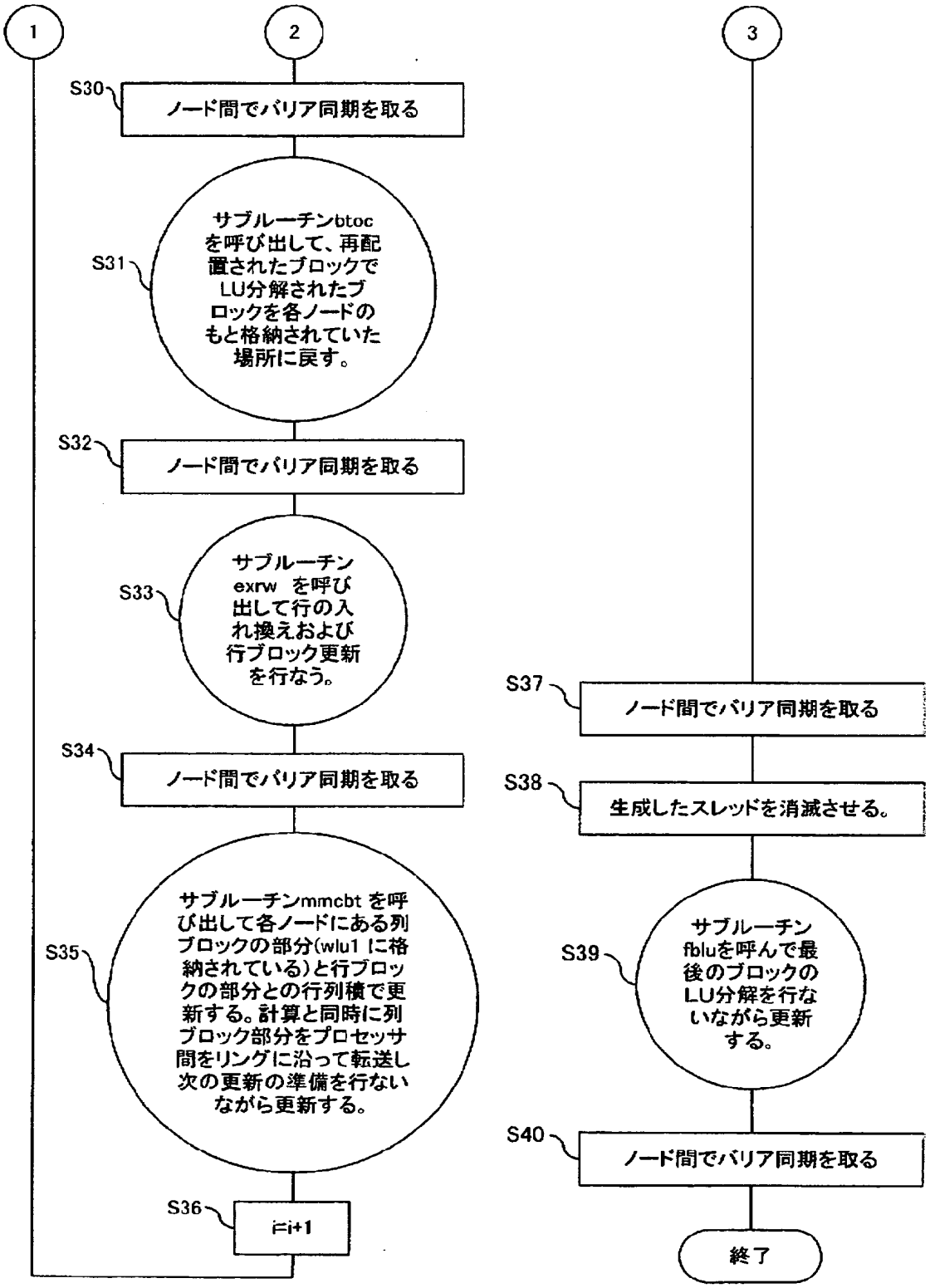
【図１２】

本発明の実施形態のフローチャート（その１）



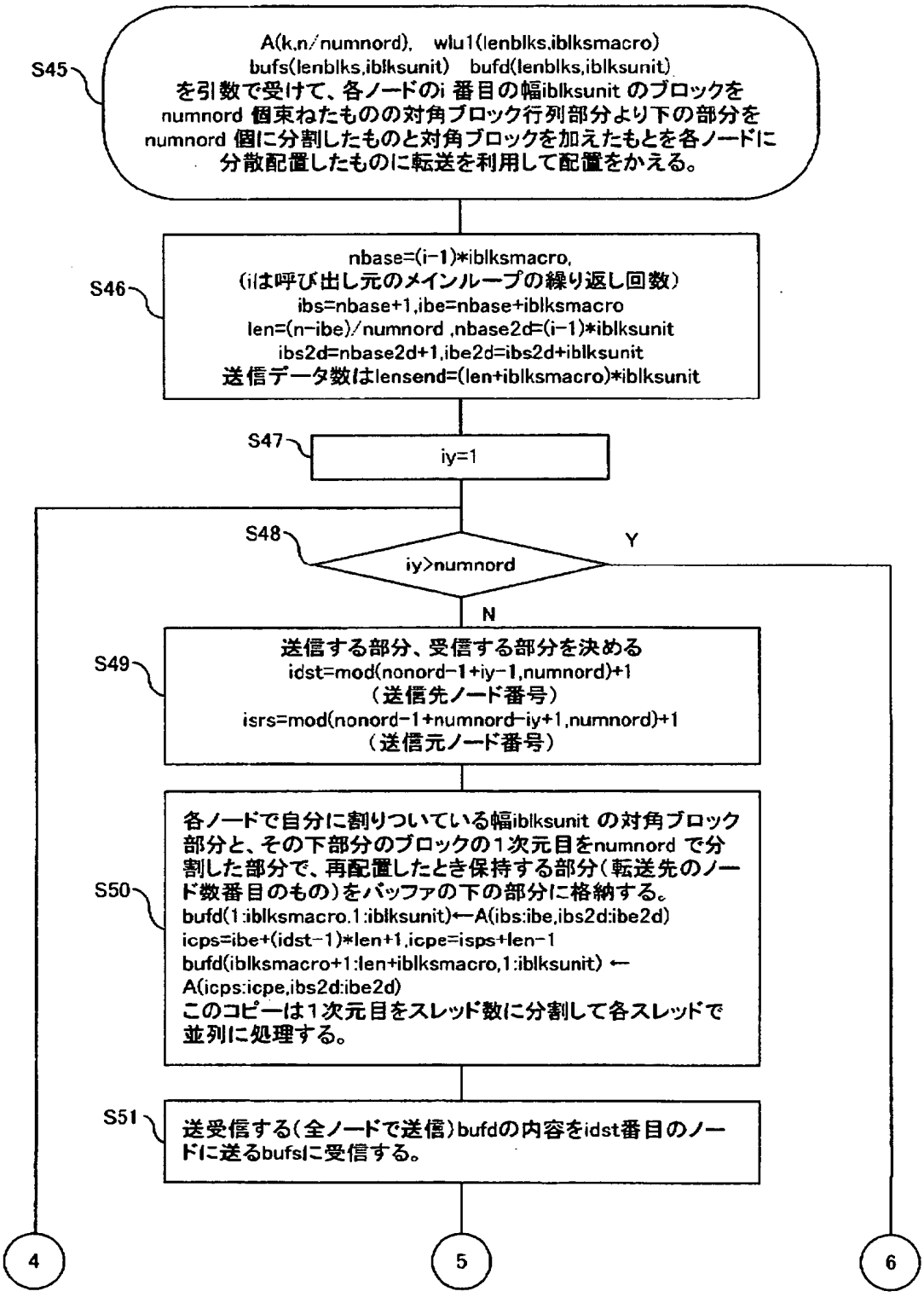
【図１３】

本発明の実施形態のフローチャート(その2)



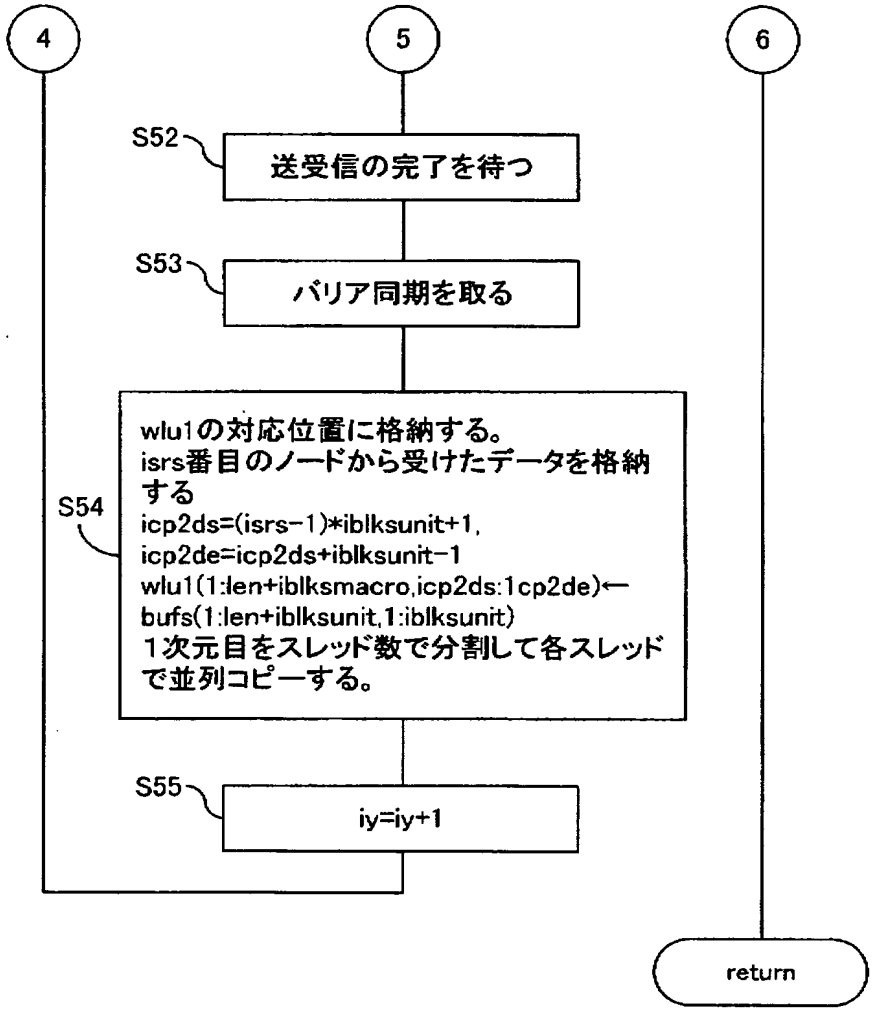
【図１４】

本発明の実施形態のフローチャート(その3)



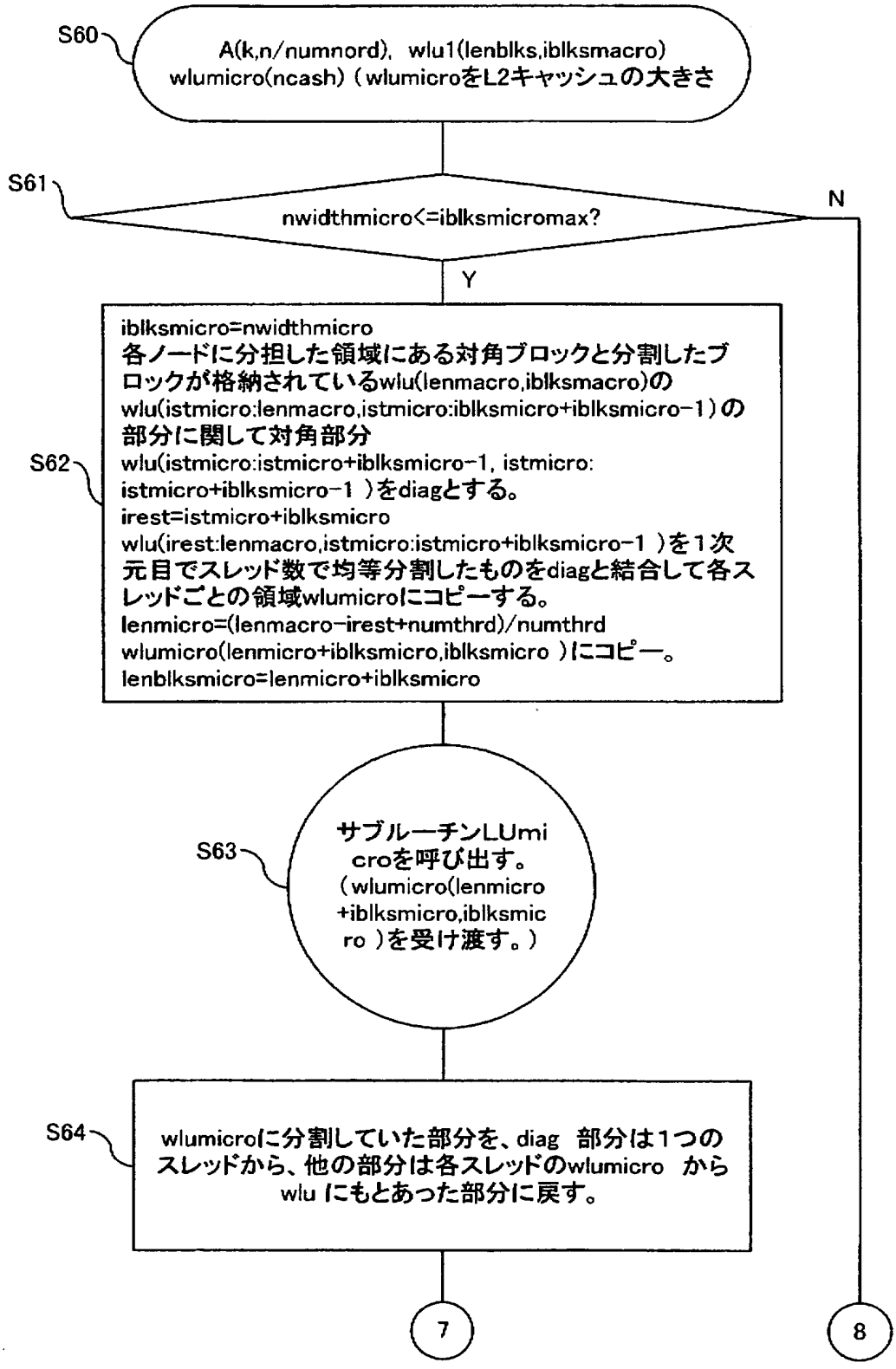
【図１５】

本発明の実施形態のフローチャート(その4)



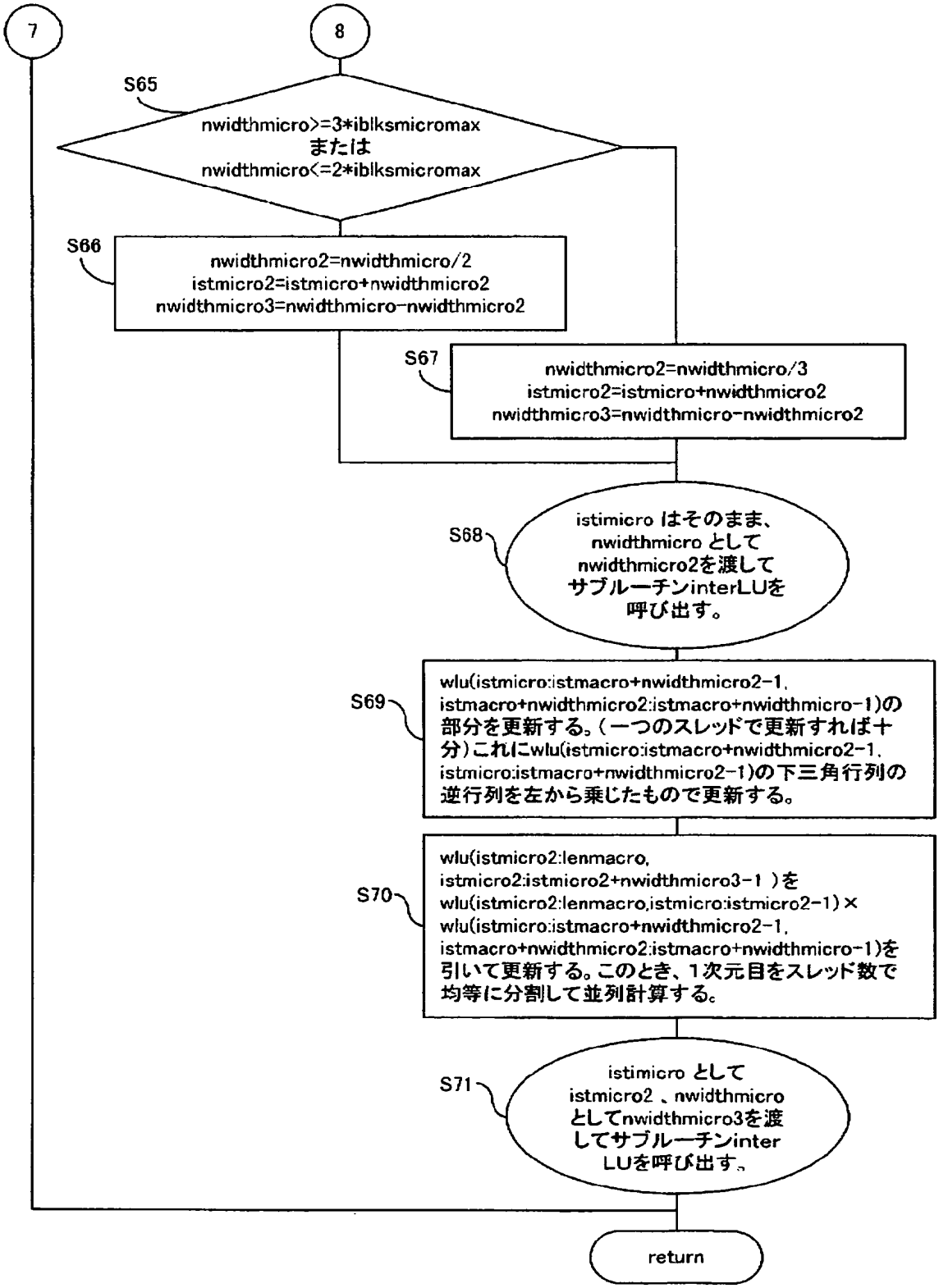
【図１６】

本発明の実施形態のフローチャート(その5)



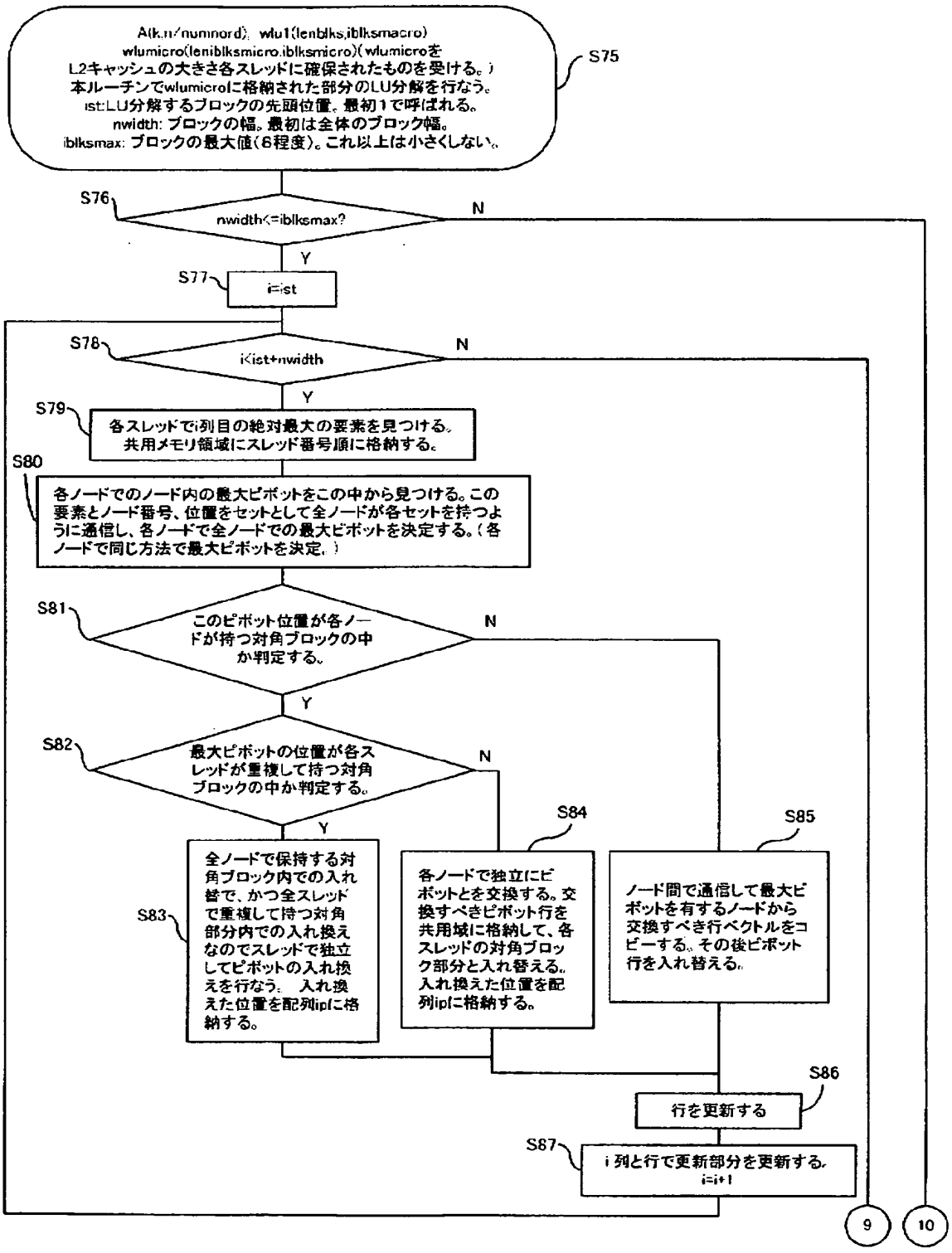
【図１７】

本発明の実施形態のフローチャート(その6)



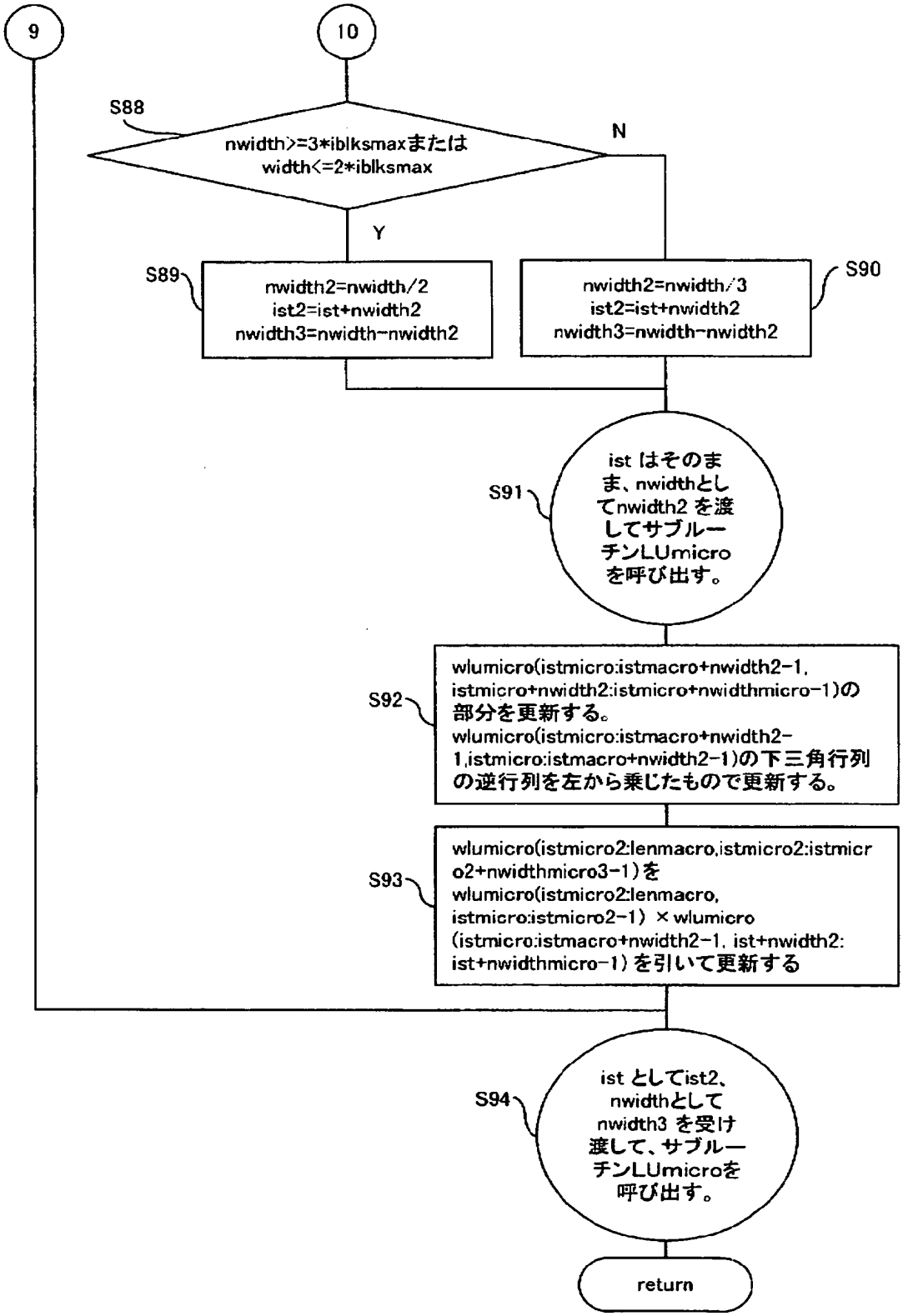
【図１８】

本発明の実施形態のフローチャート(その7)



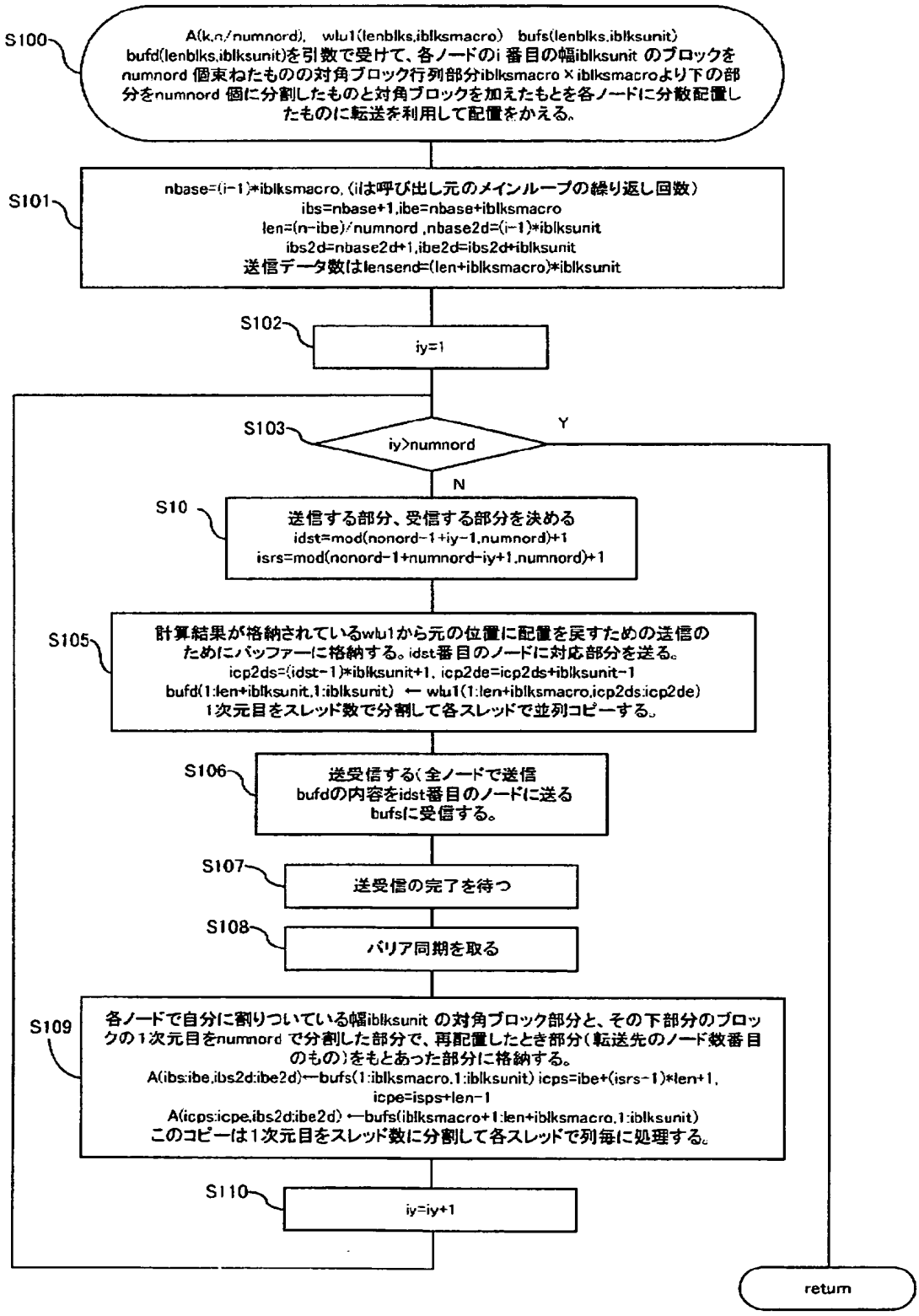
【図１９】

本発明の実施形態のフローチャート(その8)



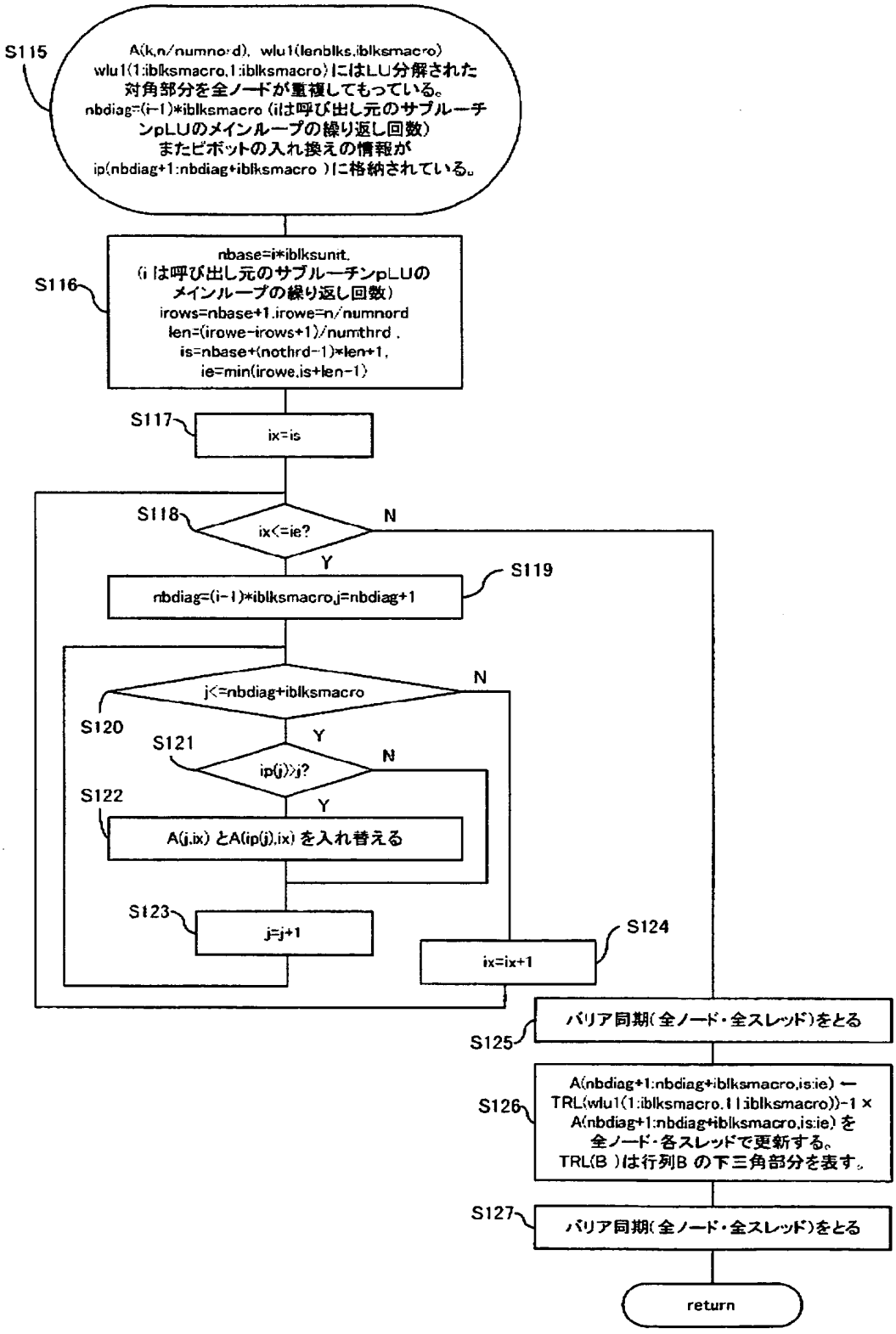
【図２０】

本発明の実施形態のフローチャート(その9)



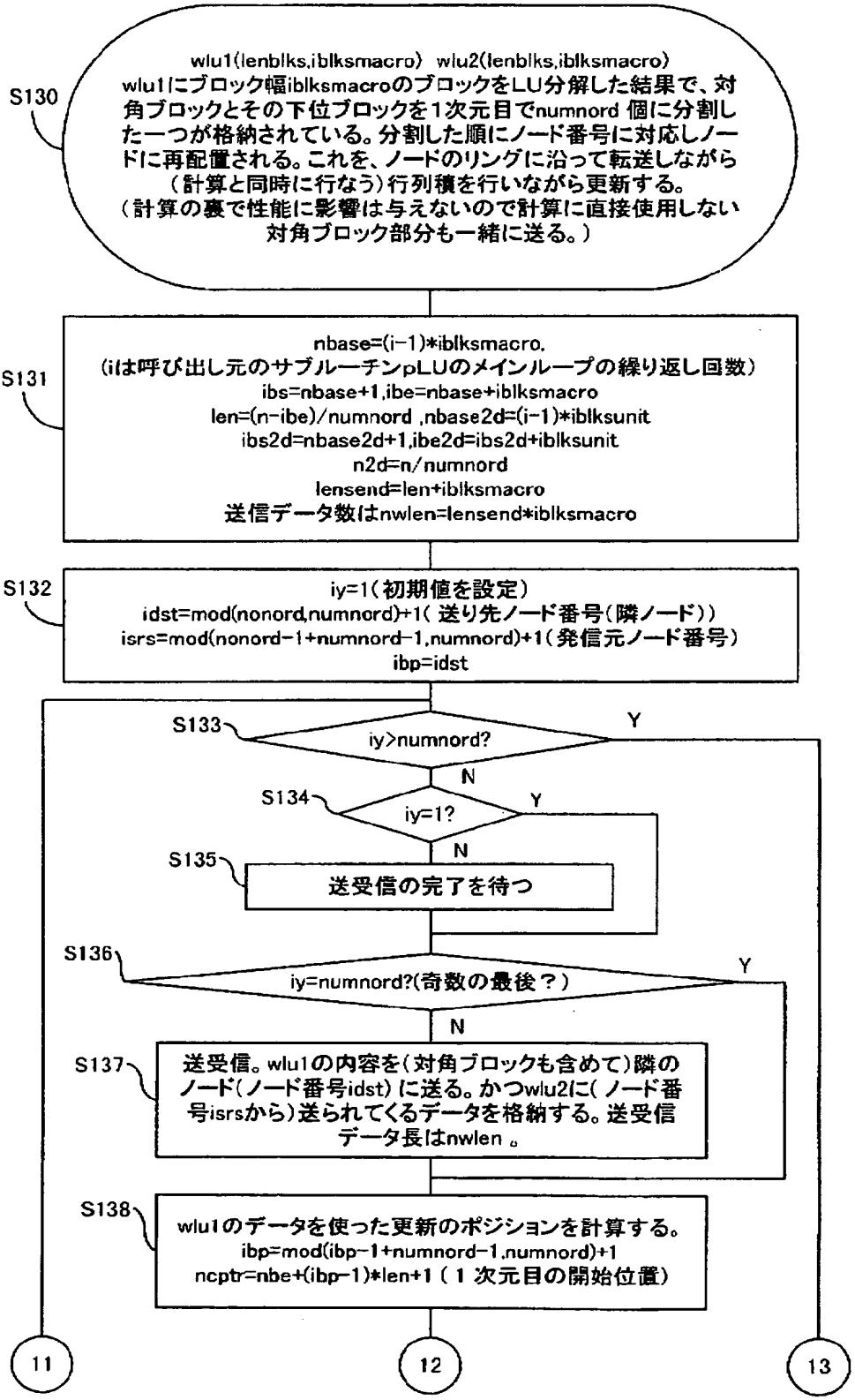
【図２１】

本発明の実施形態のフローチャート(その10)



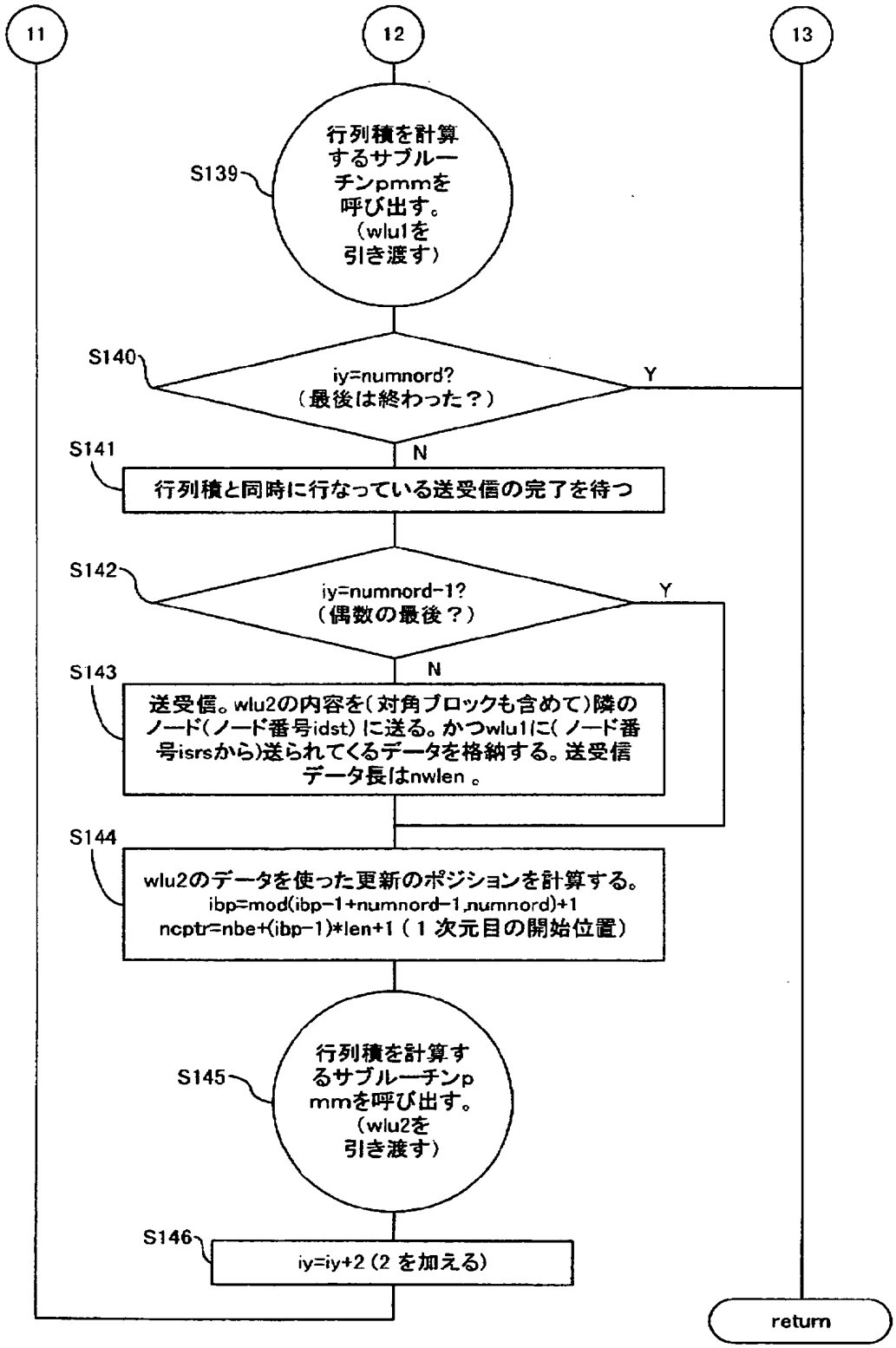
【図２２】

本発明の実施形態のフローチャート(その11)



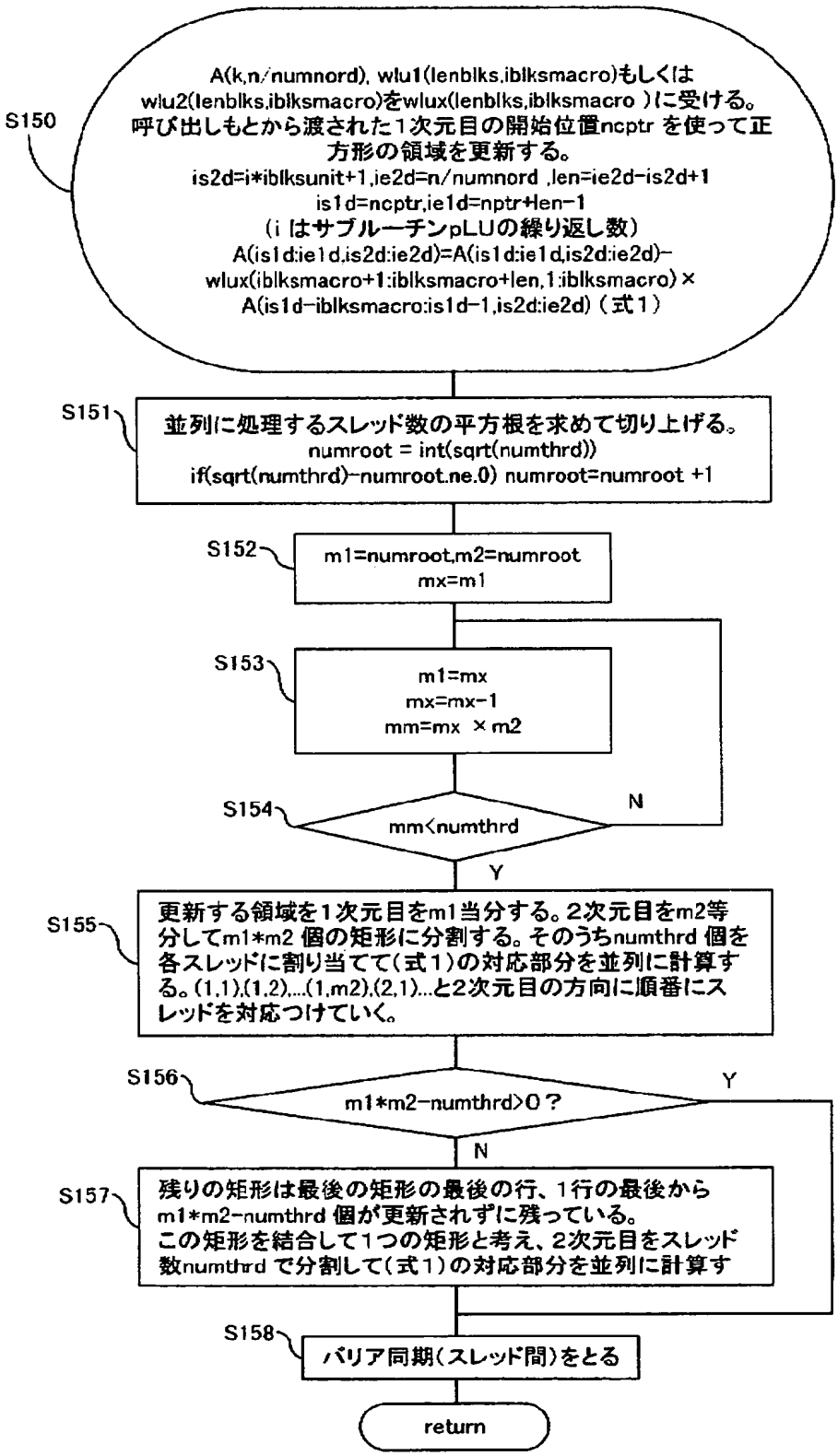
【図２３】

本発明の実施形態のフローチャート(その12)



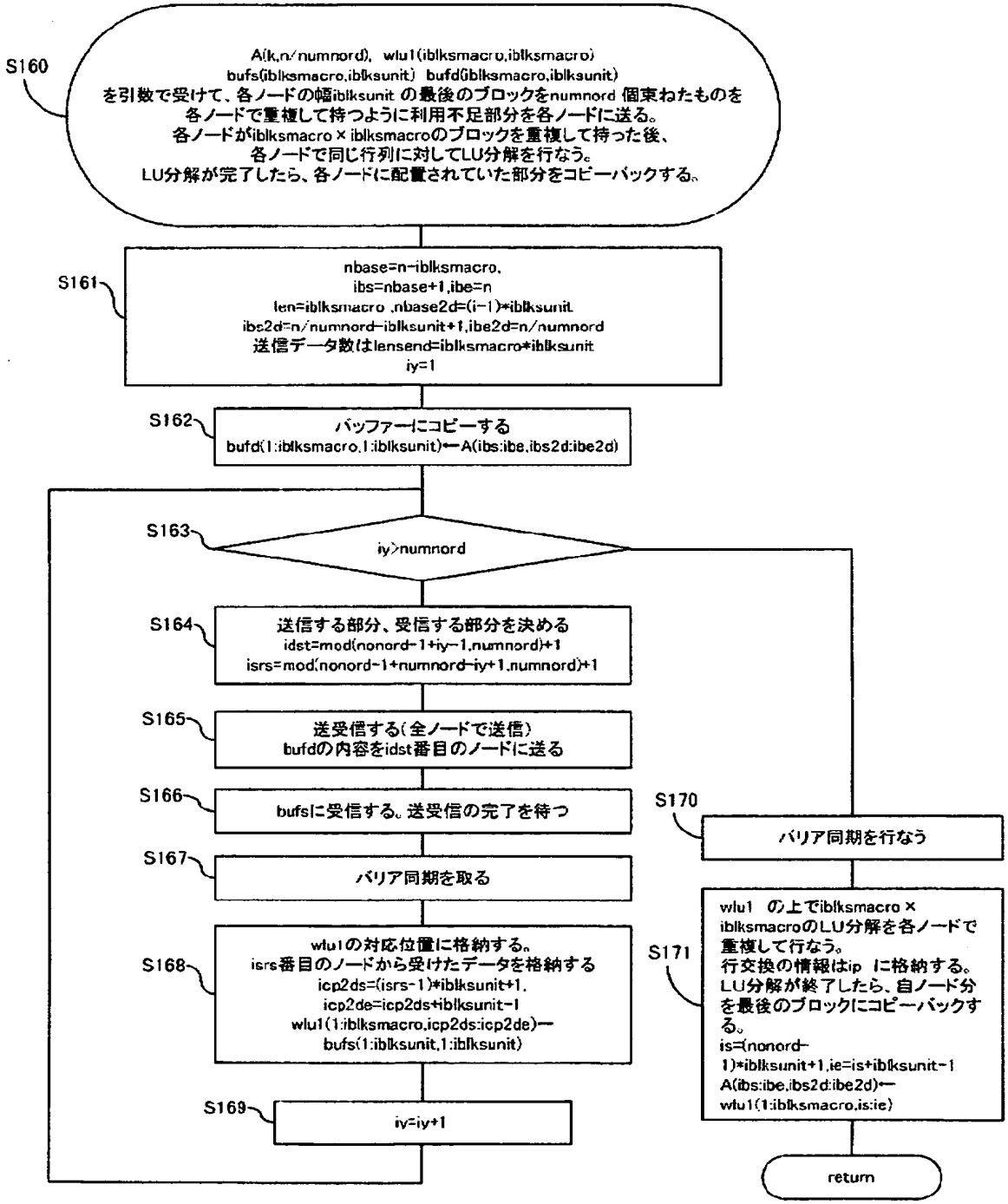
【図２４】

本発明の実施形態のフローチャート(その13)



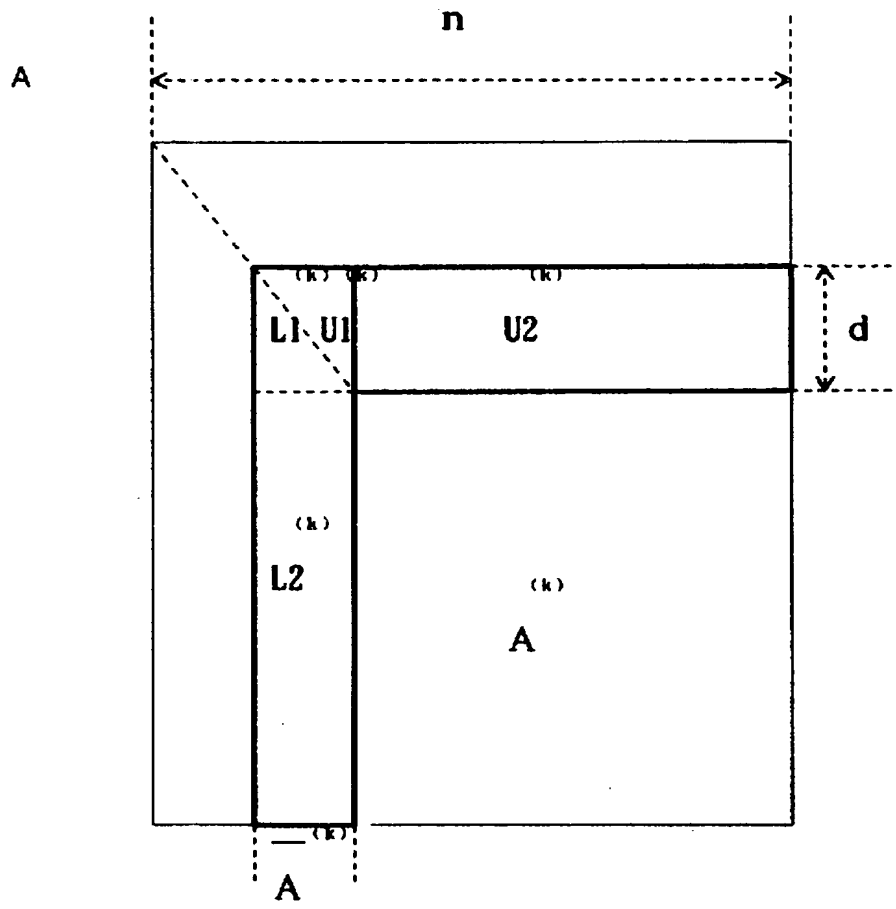
【図２５】

本発明の実施形態のフローチャート(その14)



【図２６】

スーパスカラ並列計算機用LU分解法のアルゴリズムを概略説明する図



【書類名】　　要約書

【要約】

【課題】ＳＭＰノード分散メモリ型並列計算機で高速に行列を処理することの出来る装置あるいは方法を提供する。

【解決手段】ブロック化された行列のＬＵ分解法において、ネットワークで接続されたＳＭＰノードのそれぞれに、行列の内、更新べきブロックを縦にノード数分に分割し、それぞれ分割された部分を各ノードに配置する。このような配置を更新すべきブロックについて、続けて行い、各ノードに、分割されたブロックの分割部分をサイクリックに割り当てるようにする。割り当てられた各ノードは、元のブロックの順番で分割されたブロックを更新する。元のブロックを順次更新することにより、各ノードの処理済みブロックの量が同じ分ずつ増えていき、負荷の分散をすることができる。

【選択図】　　図４

特願２００３－０９５７２０

出　願　人　履　歴　情　報

識別番号　　［０００００５２２３］

１．変更年月日　　１９９６年　３月２６日
［変更理由］　　住所変更
住　所　　神奈川県川崎市中原区上小田中４丁目１番１号
氏　名　　富士通株式会社